Skelly GoGo

Generated by Doxygen 1.9.2

Chapter 1

Source content

This folder should contain only hpp/cpp files of your implementation. You can also place hpp files in a separate directory `include`.

You can create a summary of files here. It might be useful to describe file relations, and brief summary of their content.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 Collision Namespace Reference

Classes

- class BitmaskManager
- class OrientedBoundingBox

Functions

- bool PixelPerfectTest (const sf::Sprite &Object1, const sf::Sprite &Object2, sf::Uint8 AlphaLimit)
- bool CreateTextureAndBitmask (sf::Texture &LoadInto, const std::string &Filename)
- sf::Vector2f GetSpriteCenter (const sf::Sprite &Object)
- sf::Vector2f GetSpriteSize (const sf::Sprite &Object)
- bool CircleTest (const sf::Sprite &Object1, const sf::Sprite &Object2)
- bool BoundingBoxTest (const sf::Sprite &Object1, const sf::Sprite &Object2)

Variables

- BitmaskManager Bitmasks

6.1.1 Function Documentation

6.1.1.1 BoundingBoxTest()

```
bool Collision::BoundingBoxTest (
            const sf::Sprite & Object1,
            const sf::Sprite & Object2 )
```

Test for bounding box collision using the Separating Axis Theorem Supports scaling and rotation

6.1.1.2 CircleTest()

```
bool Collision::CircleTest (
            const sf::Sprite & Object1,
            const sf::Sprite & Object2 )
```

Test for collision using circle collision dection Radius is averaged from the dimensions of the sprite so roughly circular objects will be much more accurate

6.1.1.3 CreateTextureAndBitmask()

```
bool Collision::CreateTextureAndBitmask (
            sf::Texture & LoadInto,
            const std::string & Filename )
```

Replaces Texture::loadFromFile Load an imagefile into the given texture and create a bitmask for it This is much faster than creating the bitmask for a texture on the first run of "PixelPerfectTest"

The function returns false if the file could not be opened for some reason

6.1.1.4 GetSpriteCenter()

```
sf::Vector2f Collision::GetSpriteCenter (
            const sf::Sprite & Object )
```

6.1.1.5 GetSpriteSize()

```
sf::Vector2f Collision::GetSpriteSize (
            const sf::Sprite & Object )
```

6.1.1.6 PixelPerfectTest()

```
bool Collision::PixelPerfectTest (
            const sf::Sprite & Object1,
            const sf::Sprite & Object2,
            sf::Uint8 AlphaLimit = 0 )
```

Test for a collision between two sprites by comparing the alpha values of overlapping pixels Supports scaling and rotation AlphaLimit: The threshold at which a pixel becomes "solid". If AlphaLimit is 127, a pixel with alpha value 128 will cause a collision and a pixel with alpha value 126 will not.

This functions creates bitmasks of the textures of the two sprites by downloading the textures from the graphics card to memory -> SLOW! You can avoid this by using the "CreateTextureAndBitmask" function

6.1.2 Variable Documentation

6.1.2.1 Bitmasks

```
BitmaskManager Collision::Bitmasks
```

6.2 direction Namespace Reference

Functions

- Direction GetOppositeDir (Direction direction)

6.2.1 Function Documentation

6.2.1.1 GetOppositeDir()

```
Direction direction::GetOppositeDir (
            Direction direction )
```

6.3 randomhelper Namespace Reference

Functions

- float RandomFloatBetween (float min, float max)
- int RandomIntBetween (int min, int max)

6.3.1 Function Documentation

6.3.1.1 RandomFloatBetween()

```
float randomhelper::RandomFloatBetween (
            float min,
            float max )
```

6.3.1.2 RandomIntBetween()

```
int randomhelper::RandomIntBetween (
            int min,
            int max )
```

6.4 spritehelper Namespace Reference

Functions

- void CreateSpriteFrom (const std::string &spriteFile, sf::Vector2f dimensions, sf::Sprite &sprite, sf::Texture &texture)
- void SetScale (sf::Vector2f wantedDimension, sf::Sprite &sprite)
- void RotateSprite (sf::Vector2f directionOfRotation, sf::Sprite &sprite)
- void SetOriginBottomCenter (sf::Sprite &sprite)

6.4.1 Function Documentation

6.4.1.1 CreateSpriteFrom()

```
void spritehelper::CreateSpriteFrom (
            const std::string & spriteFile,
            sf::Vector2f dimensions,
            sf::Sprite & sprite,
            sf::Texture & texture )
```

6.4.1.2 RotateSprite()

```
void spritehelper::RotateSprite (
            sf::Vector2f directionOfRotation,
            sf::Sprite & sprite )
```

6.4.1.3 SetOriginBottomCenter()

```
void spritehelper::SetOriginBottomCenter (
            sf::Sprite & sprite )
```

6.4.1.4 SetScale()

```
void spritehelper::SetScale (
            sf::Vector2f wantedDimension,
            sf::Sprite & sprite )
```

6.5 util Namespace Reference

Classes

- struct IPlatform
- struct LinuxPlatform
- struct WindowsPlatform

Chapter 7

Class Documentation

7.1 Animation Class Reference

Animation build the characters animation frames from spritesheet and displays them to achive an animated sprite.

`#include <animation.hpp>`

Public Member Functions

- Animation ()=default
- Animation (int x, int y, int width, int height, int spacing, const std::string &textureName)

  Construct a new Animation object.
- ~Animation ()
- void AnimationToSprite (sf::Sprite &sprite) const

  sets the right texture to sprite
- void Update (float dt)

  updates the sprite based on time passed

7.1.1 Detailed Description

Animation build the characters animation frames from spritesheet and displays them to achive an animated sprite.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 Animation() [1/2]

`Animation::Animation ( )  [default]`

7.1.2.2 Animation() [2/2]

```
Animation::Animation (
            int x,
            int y,
            int width,
            int height,
            int spacing,
            const std::string & textureName )
```

Construct a new Animation object.

Parameters

x	x-coordinate of rectangle defining the sprite in spritesheet
y	y-coordinate of rectangle defining the sprite in spritesheet
width	width of the character in animation
height	height of the character in animation
textureName	the spritesheet animation pulls from to create the animation

7.1.2.3 ~Animation()

```
Animation::~Animation ( )  [inline]
```

7.1.3 Member Function Documentation

7.1.3.1 AnimationToSprite()

```
void Animation::AnimationToSprite (
            sf::Sprite & sprite ) const
```

sets the right texture to sprite

Parameters

sprite	sprite which the texuture is added to

7.1.3.2 Update()

```
void Animation::Update (
            float dt )
```

updates the sprite based on time passed

Parameters

dt	time parameter

The documentation for this class was generated from the following files:

- src/Animation/animation.hpp
- src/Animation/animation.cpp

7.2 AnimationHandler Class Reference

`#include <Animationhandler.hpp>`

Public Member Functions

- AnimationHandler ()
- ~AnimationHandler ()
- AnimationHandler (uint xOffset, uint yOffset, uint width, uint height, uint xSpacing, const std::string &textureLocation, const std::string &deathTexture)
- void setAnimation (AnimationIndex index)
- Animation * getAnimation () const

7.2.1 Constructor & Destructor Documentation

7.2.1.1 AnimationHandler() [1/2]

```
AnimationHandler::AnimationHandler ( )  [inline]
```

7.2.1.2 ~AnimationHandler()

```
AnimationHandler::~AnimationHandler ( )
```

7.2.1.3 AnimationHandler() [2/2]

```
AnimationHandler::AnimationHandler (
            uint xOffset,
            uint yOffset,
            uint width,
            uint height,
            uint xSpacing,
            const std::string & textureLocation,
            const std::string & deathTexture )
```

7.2.2 Member Function Documentation

7.2.2.1 getAnimation()

```
Animation * AnimationHandler::getAnimation ( ) const
```

7.2.2.2 setAnimation()

```
void AnimationHandler::setAnimation (
            AnimationIndex index )
```

The documentation for this class was generated from the following files:

- src/Animation/Animationhandler.hpp
- src/Animation/Animationhandler.cpp

7.3 Collision::BitmaskManager Class Reference

Public Member Functions

- ~BitmaskManager ()
- sf::Uint8 GetPixel (const sf::Uint8 *mask, const sf::Texture *tex, unsigned int x, unsigned int y)
- sf::Uint8 * GetMask (const sf::Texture *tex)
- sf::Uint8 * CreateMask (const sf::Texture *tex, const sf::Image &img)

7.3.1 Constructor & Destructor Documentation

7.3.1.1 ~BitmaskManager()

```
Collision::BitmaskManager::~BitmaskManager ( )  [inline]
```

7.3.2 Member Function Documentation

7.3.2.1 CreateMask()

```
sf::Uint8 * Collision::BitmaskManager::CreateMask (
            const sf::Texture * tex,
            const sf::Image & img )  [inline]
```

7.3.2.2 GetMask()

```
sf::Uint8 * Collision::BitmaskManager::GetMask (
            const sf::Texture * tex )  [inline]
```

7.3.2.3 GetPixel()

```
sf::Uint8 Collision::BitmaskManager::GetPixel (
            const sf::Uint8 * mask,
            const sf::Texture * tex,
            unsigned int x,
            unsigned int y )  [inline]
```

The documentation for this class was generated from the following file:

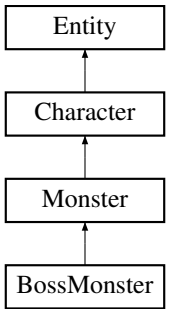
- src/Utility/Collision.cpp

7.4 BossMonster Class Reference

This is the boss monster. It is the most powerful monster and once the player kills it the game is won.

`#include <BossMonster.hpp>`

Inheritance diagram for BossMonster:



Public Member Functions

- BossMonster (PlayerPS player, float xPos, float yPos)
- BossMonster (PlayerPS player, sf::Vector2f pos)
- ~BossMonster ()
- virtual std::list< ProjectileUP > Attack ()
- virtual bool Move (float dt)
- void initVariables ()

Additional Inherited Members

7.4.1 Detailed Description

This is the boss monster. It is the most powerful monster and once the player kills it the game is won.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 BossMonster() [1/2]

```
BossMonster::BossMonster (
            PlayerPS player,
            float xPos,
            float yPos )
```

7.4.2.2 BossMonster() [2/2]

```
BossMonster::BossMonster (
            PlayerPS player,
            sf::Vector2f pos )
```

7.4.2.3 ~BossMonster()

```
BossMonster::~BossMonster ( )
```

7.4.3 Member Function Documentation

7.4.3.1 Attack()

```
std::list< ProjectileUP > BossMonster::Attack ( )  [virtual]
```

Implements Monster.

7.4.3.2 initVariables()

```
void BossMonster::initVariables ( )
```

7.4.3.3 Move()

```
bool BossMonster::Move (
            float dt )  [virtual]
```

Implements Monster.

The documentation for this class was generated from the following files:

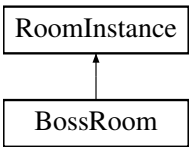
- src/Actors/Monsters/BossMonster.hpp
- src/Actors/Monsters/BossMonster.cpp

7.5 BossRoom Class Reference

The roominstance where the boss spawns and which has to be cleared to beat the game.

```
#include <BossRoom.hpp>
```

Inheritance diagram for BossRoom:



Public Member Functions

- BossRoom (sf::Vector2u window_size, sf::Vector2i coords)
- BossRoom ()
- ~BossRoom ()
- virtual void Enter (PlayerPS player, Direction direction)
- virtual void setTiles (sf::Vector2u window_size)

Additional Inherited Members

7.5.1 Detailed Description

The roominstance where the boss spawns and which has to be cleared to beat the game.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 BossRoom() [1/2]

```
BossRoom::BossRoom (
            sf::Vector2u window_size,
            sf::Vector2i coords )
```

See also

RoomInstance::RoomInstance(sf::Vector2u window_size, sf::Vector2i coords)

7.5.2.2 BossRoom() [2/2]

```
BossRoom::BossRoom ( )  [inline]
```

7.5.2.3 ~BossRoom()

```
BossRoom::~BossRoom ( )  [inline]
```

7.5.3 Member Function Documentation

7.5.3.1 Enter()

```
void BossRoom::Enter (
            PlayerPS player,
            Direction direction )  [virtual]
```

Parameters

player	Desciption
direction	Desciption

Reimplemented from RoomInstance.

7.5.3.2 setTiles()

```
void BossRoom::setTiles (
            sf::Vector2u window_size )  [virtual]
```

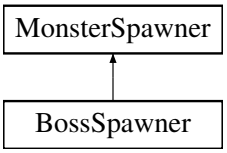
The documentation for this class was generated from the following files:

- src/Dungeon/specialrooms/BossRoom.hpp
- src/Dungeon/specialrooms/BossRoom.cpp

7.6 BossSpawner Class Reference

```
#include <BossSpawner.hpp>
```

Inheritance diagram for BossSpawner:



Public Member Functions

- BossSpawner ()
- ~BossSpawner ()
- virtual MonsterSP SpawnMonster (sf::Vector2u roomSize, PlayerPS target)

Additional Inherited Members

7.6.1 Constructor & Destructor Documentation

7.6.1.1 BossSpawner()

```
BossSpawner::BossSpawner ( )  [inline]
```

7.6.1.2 ~BossSpawner()

```
BossSpawner::~BossSpawner ( )  [inline]
```

7.6.2 Member Function Documentation

7.6.2.1 SpawnMonster()

```
MonsterSP BossSpawner::SpawnMonster (
            sf::Vector2u roomSize,
            PlayerPS target )  [virtual]
```

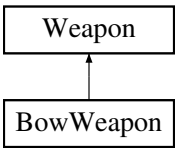
The documentation for this class was generated from the following files:

- src/Actors/Monsters/MonsterSpawner/BossSpawner.hpp
- src/Actors/Monsters/MonsterSpawner/BossSpawner.cpp

7.7 BowWeapon Class Reference

```
#include <BowWeapon.hpp>
```

Inheritance diagram for BowWeapon:



Public Member Functions

- BowWeapon (int damage, int range, int rateOfFire, float projectileSpeed, Vector2f projectileSize, const std↩ ::string &spriteLocation)
- virtual ~BowWeapon ()
- virtual ProjectileUP Use (Vector2f dir, Vector2f origin)

Additional Inherited Members

7.7.1 Constructor & Destructor Documentation

7.7.1.1 BowWeapon()

```
BowWeapon::BowWeapon (
            int damage,
            int range,
            int rateOfFire,
            float projectileSpeed,
            Vector2f projectileSize,
            const std::string & spriteLocation )
```

7.7.1.2 ~BowWeapon()

```
virtual BowWeapon::~BowWeapon ( )  [inline], [virtual]
```

7.7.2 Member Function Documentation

7.7.2.1 Use()

```
ProjectileUP BowWeapon::Use (
            Vector2f dir,
            Vector2f origin )  [virtual]
```

Implements Weapon.

The documentation for this class was generated from the following files:

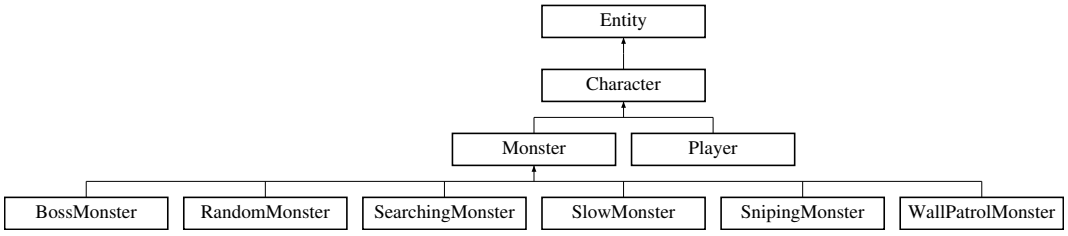
- src/Combat/Weapons/BowWeapon.hpp
- src/Combat/Weapons/BowWeapon.cpp

7.8 Character Class Reference

Character class that our player and monster inherits.

`#include <character.hpp>`

Inheritance diagram for Character:



Public Member Functions

- Character (const std::string &filename, sf::Vector2f pos, bool animated=false)
- virtual ~Character ()
- virtual void Update (float dt)=0
- void Equip (Weapon *weapon)

  Equips weapon to character.
- void initVariables ()
- void TakeDamage (int value)

  removes hitpoints corresponding to the damage taken value
- void Heal (int value)

  Heal character.
- int GetHitPoints () const
- bool IsAlive ()

  Check if player is alive.
- bool HasWeapon ()

  Check if player has weapon equipped.
- bool Idle ()

  Set animation to IDLE.
- bool Dead ()

  Set animation to DEAD.
- bool MoveLeft (float dt)

  Sets the animation to right direction (LEFT) and moves the character.
- bool MoveRight (float dt)

  Sets the animation to right direction (RIGHT) and moves the character.
- bool MoveDown (float dt)

  Sets the animation to right direction (DOWN) and moves the character.
- bool MoveUp (float dt)

  Sets the animation to right direction (UP) and moves the character.
- virtual bool Move (float)
- void RevertMove ()

  revert movement to last position, used when hit wall
- void ResetAttackCooldown ()
- float GetAttackCooldownLeft () const
- float GetAttackCooldownLength () const
- int GetMaxHP ()
- void SetNormalSpeed (float value)
- void ResetCharacterToBeAlive ()

Public Attributes

- bool CanAttack

Protected Member Functions

- void generalUpdate (float dt)
- void updateAttackCooldown (float dt)
- std::list< ProjectileUP > emptyList ()
- std::list< ProjectileUP > shotProjectileList (sf::Vector2f aimPos)

Protected Attributes

- Weapon * weapon_
- Projectile::Type characterProjectileType_
- sf::Vector2f startPos
- int hitpoints_
- int currentMaxHitpoints_
- int defaultMaxHitpoints_
- bool hasAnimation_
- AnimationHandler animationHandler_
- float currentSpeed_
- float defaultSpeed_
- bool left_or_right_ = true
- bool invincibility_frame_ = false
- float attackCooldownLength_
- float attackCooldownLeft

7.8.1 Detailed Description

Character class that our player and monster inherits.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 Character()

```
Character::Character (
            const std::string & filename,
            sf::Vector2f pos,
            bool animated = false )
```

7.8.2.2 ~Character()

```
Character::~Character ( )  [virtual]
```

7.8.3 Member Function Documentation

7.8.3.1 Dead()

```
bool Character::Dead ( )
```

Set animation to DEAD.

Returns

true when set

7.8.3.2 emptyList()

```
std::list< ProjectileUP > Character::emptyList ( )  [protected]
```

7.8.3.3 Equip()

```
void Character::Equip (
            Weapon * weapon )
```

Equips weapon to character.

Parameters

weapon	weapon to be equipped

7.8.3.4 generalUpdate()

```
void Character::generalUpdate (
            float dt )  [protected]
```

7.8.3.5 GetAttackCooldownLeft()

```
float Character::GetAttackCooldownLeft ( ) const  [inline]
```

7.8.3.6 GetAttackCooldownLength()

```
float Character::GetAttackCooldownLength ( ) const  [inline]
```

7.8.3.7 GetHitPoints()

```
int Character::GetHitPoints ( ) const
```

7.8.3.8 GetMaxHP()

```
int Character::GetMaxHP ( )
```

7.8.3.9 HasWeapon()

```
bool Character::HasWeapon ( )
```

Check if player has weapon equipped.

Returns

true if has

false if not

7.8.3.10 Heal()

```
void Character::Heal (
            int value )
```

Heal character.

Parameters

value	value healed

7.8.3.11 Idle()

```
bool Character::Idle ( )
```

Set animation to IDLE.

Returns

true when set

7.8.3.12 initVariables()

```
void Character::initVariables ( )
```

7.8.3.13 IsAlive()

```
bool Character::IsAlive ( )
```

Check if player is alive.

Returns

true if alive

false if not

7.8.3.14 Move()

```
virtual bool Character::Move (
            float  )  [inline], [virtual]
```

Reimplemented in BossMonster, RandomMonster, SearchingMonster, SlowMonster, SnipingMonster, WallPatrolMonster, and Monster.

7.8.3.15 MoveDown()

```
bool Character::MoveDown (
            float dt )
```

Sets the animation to right direction (DOWN) and moves the character.

Parameters

dt	deltatime

Returns

true when set

7.8.3.16 MoveLeft()

```
bool Character::MoveLeft (
            float dt )
```

Sets the animation to right direction (LEFT) and moves the character.

Parameters

dt	deltatime

Returns

true when set

else if (!left_or_right_) { sprite_.setScale(-sprite_.getScale().x, sprite_.getScale().y); sprite_.setPosition(pos_.x + sprite_.getLocalBounds().width, pos_.y); left_or_right_ = true; }

7.8.3.17 MoveRight()

```
bool Character::MoveRight (
            float dt )
```

Sets the animation to right direction (RIGHT) and moves the character.

Parameters

dt	deltatime

Returns

true when set

else if (left_or_right_) { sprite_.setScale(sprite_.getScale().x, sprite_.getScale().y); sprite_.setPosition(pos_.x + 64, pos_.y); left_or_right_ = false; }

7.8.3.18 MoveUp()

```
bool Character::MoveUp (
            float dt )
```

Sets the animation to right direction (UP) and moves the character.

Parameters

dt	deltatime

Returns

true when set

7.8.3.19 ResetAttackCooldown()

```
void Character::ResetAttackCooldown ( )
```

7.8.3.20 ResetCharacterToBeAlive()

```
void Character::ResetCharacterToBeAlive ( )
```

7.8.3.21 RevertMove()

```
void Character::RevertMove ( )
```

revert movement to last position, used when hit wall

7.8.3.22 SetNormalSpeed()

```
void Character::SetNormalSpeed (
            float value )
```

7.8.3.23 shotProjectileList()

```
std::list< ProjectileUP > Character::shotProjectileList (
            sf::Vector2f aimPos )  [protected]
```

7.8.3.24 TakeDamage()

```
void Character::TakeDamage (
            int value )
```

removes hitpoints corresponding to the damage taken value

Parameters

value	damage taken

7.8.3.25 Update()

```
virtual void Character::Update (
            float dt )  [pure virtual]
```

Implemented in Player, and Monster.

7.8.3.26 updateAttackCooldown()

```
void Character::updateAttackCooldown (
            float dt )  [protected]
```

7.8.4 Member Data Documentation

7.8.4.1 animationHandler_

```
AnimationHandler Character::animationHandler_  [protected]
```

7.8.4.2 attackCooldownLeft

```
float Character::attackCooldownLeft  [protected]
```

7.8.4.3 attackCooldownLength_

```
float Character::attackCooldownLength_  [protected]
```

7.8.4.4 CanAttack

```
bool Character::CanAttack
```

7.8.4.5 characterProjectileType_

```
Projectile::Type Character::characterProjectileType_  [protected]
```

7.8.4.6 currentMaxHitpoints_

```
int Character::currentMaxHitpoints_  [protected]
```

7.8.4.7 currentSpeed_

```
float Character::currentSpeed_  [protected]
```

7.8.4.8 defaultMaxHitpoints_

```
int Character::defaultMaxHitpoints_  [protected]
```

7.8.4.9 defaultSpeed_

```
float Character::defaultSpeed_  [protected]
```

7.8.4.10 hasAnimation_

```
bool Character::hasAnimation_  [protected]
```

7.8.4.11 hitpoints_

```
int Character::hitpoints_  [protected]
```

7.8.4.12 invincibility_frame_

```
bool Character::invincibility_frame_ = false  [protected]
```

7.8.4.13 left_or_right_

```
bool Character::left_or_right_ = true  [protected]
```

7.8.4.14 startPos

```
sf::Vector2f Character::startPos  [protected]
```

7.8.4.15 weapon_

```
Weapon* Character::weapon_  [protected]
```

The documentation for this class was generated from the following files:

- src/Actors/character.hpp
- src/Actors/character.cpp

7.9 CollisionSystem Class Reference

Unused, was not intergrated or completed.

```
#include <CollisionSystem.hpp>
```

Public Member Functions

- CollisionSystem ()
- void AddToCollisionList (ICollidable *obj)
- void RemoveFromCollisionList (ICollidable *obj)
- void ProcessCollisionList ()

7.9.1 Detailed Description

Unused, was not intergrated or completed.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 CollisionSystem()

```
CollisionSystem::CollisionSystem ( )  [inline]
```

7.9.3 Member Function Documentation

7.9.3.1 AddToCollisionList()

```
void CollisionSystem::AddToCollisionList (
            ICollidable * obj )
```

7.9.3.2 ProcessCollisionList()

```
void CollisionSystem::ProcessCollisionList ( )
```

7.9.3.3 RemoveFromCollisionList()

```
void CollisionSystem::RemoveFromCollisionList (
            ICollidable * obj )
```

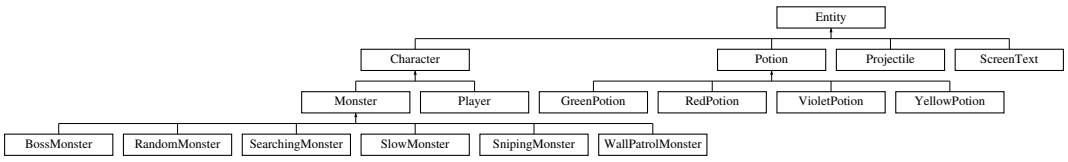
The documentation for this class was generated from the following file:

- src/Interfaces/CollisionSystem.hpp

7.10 Entity Class Reference

```
#include <entity.hpp>
```

Inheritance diagram for Entity:



Public Member Functions

- Entity (const std::string &spriteLocation, float xPos, float yPos, sf::Vector2f spriteDims)

  Construct a new game Entity.
- Entity (const std::string &spirteLocation, sf::Vector2f pos, sf::Vector2f spriteDims)
- Entity (sf::Sprite &sprite, float xPos, float yPos)
- Entity (sf::Sprite &sprite, sf::Vector2f pos)
- virtual ~Entity ()
- const sf::Sprite & GetSprite () const

  Get the entity's sprite.
- const sf::Vector2f & GetPos () const

  Get the entity position as a Vector2f.
- const sf::Vector2i GetPosI ()

  Get the entity postions.
- sf::Vector2f GetSpritePosition () const

  Get the position of the entitys sprite.
- sf::Vector2f GetSpriteCenter () const

  Get the position of the entitys sprite center.
- const sf::Vector2f & GetOldPosition () const

  Get the position of the Enitys position during the last game tick.
- sf::FloatRect GetSpriteBounds () const
- sf::FloatRect GetBaseBoxAt (sf::Vector2f pos) const
- void SetPos (sf::Vector2f pos)

  Function to set the position and sprite position of the entity.
- void SetPosAndOldPos (sf::Vector2f pos)

  Function to set both the oldPos_, pos_ and sprite position of the entity.
- virtual void Render (sf::RenderTarget *target)

Protected Member Functions

- void initSprite (const std::string &spriteLocation, sf::Vector2f spriteDims)

Protected Attributes

- sf::Vector2f pos_
- sf::Vector2f oldPos_
- sf::Sprite sprite_
- sf::Texture texture_

7.10.1 Constructor & Destructor Documentation

7.10.1.1 Entity() [1/4]

```
Entity::Entity (
            const std::string & spriteLocation,
            float xPos,
            float yPos,
            sf::Vector2f spriteDims )
```

Construct a new game Entity.

Parameters

spriteLocation	path to the sprite
xPos	x-choord for left side
yPos	y-choord for top
spriteDims	!scaling of sprite!

7.10.1.2 Entity() [2/4]

```
Entity::Entity (
            const std::string & spirteLocation,
            sf::Vector2f pos,
            sf::Vector2f spriteDims )
```

7.10.1.3 Entity() [3/4]

```
Entity::Entity (
            sf::Sprite & sprite,
            float xPos,
            float yPos )
```

7.10.1.4 Entity() [4/4]

```
Entity::Entity (
            sf::Sprite & sprite,
            sf::Vector2f pos )
```

7.10.1.5 ∼Entity()

```
virtual Entity::∼Entity ( )  [inline], [virtual]
```

7.10.2 Member Function Documentation

7.10.2.1 GetBaseBoxAt()

```
sf::FloatRect Entity::GetBaseBoxAt (
            sf::Vector2f pos ) const
```

7.10.2.2 GetOldPosition()

```
const sf::Vector2f & Entity::GetOldPosition ( ) const
```

Get the position of the Enitys position during the last game tick.

Returns

const sf::Vector2f&

7.10.2.3 GetPos()

```
const sf::Vector2f & Entity::GetPos ( ) const  [inline]
```

Get the entity position as a Vector2f.

Returns

const sf::Vector2f&

7.10.2.4 GetPosI()

```
const sf::Vector2i Entity::GetPosI ( )  [inline]
```

Get the entity postions.

Returns

const sf::Vector2i

7.10.2.5 GetSprite()

```
const sf::Sprite & Entity::GetSprite ( ) const  [inline]
```

Get the entity's sprite.

Returns

const sf::Sprite&

7.10.2.6 GetSpriteBounds()

```
sf::FloatRect Entity::GetSpriteBounds ( ) const
```

7.10.2.7 GetSpriteCenter()

```
sf::Vector2f Entity::GetSpriteCenter ( ) const
```

Get the position of the entitys sprite center.

Returns

sf::Vector2f

7.10.2.8 GetSpritePosition()

```
sf::Vector2f Entity::GetSpritePosition ( ) const  [inline]
```

Get the position of the entitys sprite.

Returns

sf::Vector2f

7.10.2.9 initSprite()

```
void Entity::initSprite (
            const std::string & spriteLocation,
            sf::Vector2f spriteDims )  [protected]
```

7.10.2.10 Render()

```
void Entity::Render (
            sf::RenderTarget * target )  [virtual]
```

Reimplemented in Monster.

7.10.2.11 SetPos()

```
void Entity::SetPos (
            sf::Vector2f pos )
```

Function to set the position and sprite position of the entity.

Parameters

pos	the new position

7.10.2.12 SetPosAndOldPos()

```
void Entity::SetPosAndOldPos (
            sf::Vector2f pos )
```

Function to set both the oldPos_, pos_ and sprite position of the entity.

Parameters

pos	the new position

7.10.3 Member Data Documentation

7.10.3.1 oldPos_

```
sf::Vector2f Entity::oldPos_  [protected]
```

7.10.3.2 pos_

```
sf::Vector2f Entity::pos_  [protected]
```

7.10.3.3 sprite_

```
sf::Sprite Entity::sprite_  [protected]
```

7.10.3.4 texture_

```
sf::Texture Entity::texture_  [protected]
```

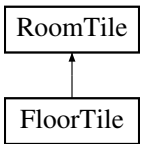
The documentation for this class was generated from the following files:

- src/entity.hpp
- src/entity.cpp

7.11 FloorTile Class Reference

```
#include <roomTile.hpp>
```

Inheritance diagram for FloorTile:



Public Member Functions

- FloorTile (std::string texture, float x, float y)

7.11.1 Constructor & Destructor Documentation

7.11.1.1 FloorTile()

```
FloorTile::FloorTile (
            std::string texture,
            float x,
            float y )  [inline]
```

The documentation for this class was generated from the following file:

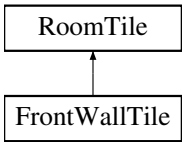
- src/Dungeon/Tiles/roomTile.hpp

7.12 FrontWallTile Class Reference

A wall that is not walkable but penetrable.

`#include <roomTile.hpp>`

Inheritance diagram for FrontWallTile:



Public Member Functions

- FrontWallTile (std::string texture, float x, float y)

7.12.1 Detailed Description

A wall that is not walkable but penetrable.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 FrontWallTile()

```
FrontWallTile::FrontWallTile (
            std::string texture,
            float x,
            float y )  [inline]
```

The documentation for this class was generated from the following file:

- src/Dungeon/Tiles/roomTile.hpp

7.13 Game Class Reference

`#include <game.hpp>`

Public Member Functions

- Game ()
- ~Game ()
- void UpdateGame ()

  Updates the game instance/variables in the game loop.
- void RenderGame ()

  Renders the updated game variables in the game loop.
- bool Running () const

  Checks that the game is running aka window is open.
- void Events ()

  pulls events such as if the window is closed

7.13.1 Constructor & Destructor Documentation

7.13.1.1 Game()

```
Game::Game ( )
```

7.13.1.2 ~Game()

```
Game::~Game ( )
```

7.13.2 Member Function Documentation

7.13.2.1 Events()

```
void Game::Events ( )
```

pulls events such as if the window is closed

7.13.2.2 RenderGame()

```
void Game::RenderGame ( )
```

Renders the updated game variables in the game loop.

7.13.2.3 Running()

```
bool Game::Running ( ) const
```

Checks that the game is running aka window is open.

Returns

true/false based if the window is open

7.13.2.4 UpdateGame()

```
void Game::UpdateGame ( )
```

Updates the game instance/variables in the game loop.

The documentation for this class was generated from the following files:

- src/game.hpp
- src/game.cpp

7.14 Gamebar Class Reference

```
#include <gamebar.hpp>
```

Public Member Functions

- Gamebar (PlayerPS player)

  Construct a new Gamebar object, meaning creates the gamebars for player health, attack cooldown and dash cooldown.
- Gamebar ()
- void Render (sf::RenderTarget *target)

  renders the gamebars
- void Update ()

  keeps the gamebars updates when damage is lost and when attack and dash is used
- void RenderInventory (sf::RenderTarget *target)

7.14.1 Constructor & Destructor Documentation

7.14.1.1 Gamebar() [1/2]

```
Gamebar::Gamebar (
            PlayerPS player )
```

Construct a new Gamebar object, meaning creates the gamebars for player health, attack cooldown and dash cooldown.

Parameters

player	player which is tracked

7.14.1.2 Gamebar() [2/2]

```
Gamebar::Gamebar ( )  [inline]
```

7.14.2 Member Function Documentation

7.14.2.1 Render()

```
void Gamebar::Render (
            sf::RenderTarget * target )
```

renders the gamebars

Parameters

target	window here gamebars should be drawn

7.14.2.2 RenderInventory()

```
void Gamebar::RenderInventory (
            sf::RenderTarget * target )
```

7.14.2.3 Update()

```
void Gamebar::Update ( )
```

keeps the gamebars updates when damage is lost and when attack and dash is used

The documentation for this class was generated from the following files:

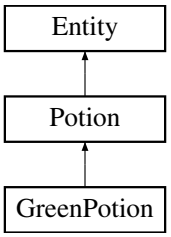
- src/gamebar.hpp
- src/gamebar.cpp

7.15 GreenPotion Class Reference

A green potion. Takes a position as parameter and the colour and healing effect is set automatically.

`#include <HealthPotions.hpp>`

Inheritance diagram for GreenPotion:



Public Member Functions

- GreenPotion (sf::Vector2f pos)

Additional Inherited Members

7.15.1 Detailed Description

A green potion. Takes a position as parameter and the colour and healing effect is set automatically.

7.15.2 Constructor & Destructor Documentation

7.15.2.1 GreenPotion()

```
GreenPotion::GreenPotion (
            sf::Vector2f pos )  [inline]
```

The documentation for this class was generated from the following file:

- src/Combat/Health/HealthPotions.hpp

7.16 ICollidable Class Reference

Unused interface for collisionsystem.

`#include <ICollidable.hpp>`

Public Types

- enum EntityType { character , projectile , tile }

Public Member Functions

- virtual sf::FloatRect GetBoundingBox ()=0
- virtual void ProcessCollision (ICollidable ∗other)=0
- virtual EntityType GetEntityType ()=0
- virtual ∼ICollidable ()

7.16.1 Detailed Description

Unused interface for collisionsystem.

7.16.2 Member Enumeration Documentation

7.16.2.1 EntityType

```
enum ICollidable::EntityType
```

Enumerator

character	
projectile	
tile	

7.16.3 Constructor & Destructor Documentation

7.16.3.1 ∼ICollidable()

```
virtual ICollidable::∼ICollidable ( )  [inline], [virtual]
```

7.16.4 Member Function Documentation

7.16.4.1 GetBoundingBox()

```
virtual sf::FloatRect ICollidable::GetBoundingBox ( )  [pure virtual]
```

7.16.4.2 GetEntityType()

```
virtual EntityType ICollidable::GetEntityType ( )  [pure virtual]
```

7.16.4.3 ProcessCollision()

```
virtual void ICollidable::ProcessCollision (
            ICollidable * other )  [pure virtual]
```

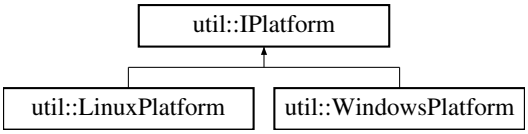
The documentation for this class was generated from the following file:

- src/Interfaces/ICollidable.hpp

7.17 util::IPlatform Struct Reference

`#include <IPlatform.hpp>`

Inheritance diagram for util::IPlatform:



Public Member Functions

- virtual ~IPlatform ()=default
- virtual void setIcon (const sf::WindowHandle &inHandle)=0
- virtual void toggleFullscreen (const sf::WindowHandle &inHandle, const sf::Uint32 inStyle, const bool in↩ Windowed, const sf::Vector2u &inResolution)=0
- virtual int getRefreshRate (const sf::WindowHandle &inHandle)=0
- virtual float getScreenScalingFactor (const sf::WindowHandle &inHandle)=0

7.17.1 Constructor & Destructor Documentation

7.17.1.1 ~IPlatform()

```
virtual util::IPlatform::~IPlatform ( )  [virtual], [default]
```

7.17.2 Member Function Documentation

7.17.2.1 getRefreshRate()

```
virtual int util::IPlatform::getRefreshRate (
            const sf::WindowHandle & inHandle )  [pure virtual]
```

Implemented in util::LinuxPlatform, and util::WindowsPlatform.

7.17.2.2 getScreenScalingFactor()

```
virtual float util::IPlatform::getScreenScalingFactor (
            const sf::WindowHandle & inHandle )  [pure virtual]
```

Implemented in util::LinuxPlatform, and util::WindowsPlatform.

7.17.2.3 setIcon()

```
virtual void util::IPlatform::setIcon (
            const sf::WindowHandle & inHandle )  [pure virtual]
```

Implemented in util::LinuxPlatform, and util::WindowsPlatform.

7.17.2.4 toggleFullscreen()

```
virtual void util::IPlatform::toggleFullscreen (
            const sf::WindowHandle & inHandle,
            const sf::Uint32 inStyle,
            const bool inWindowed,
            const sf::Vector2u & inResolution )  [pure virtual]
```

Implemented in util::LinuxPlatform, and util::WindowsPlatform.

The documentation for this struct was generated from the following file:

- src/Platform/IPlatform.hpp

7.18 LevelUpInstance Class Reference

`#include <LevelUpInstance.hpp>`

Public Member Functions

- LevelUpInstance ()

  Construct a new Level Up Instance object, designed to only be used from the LevelUpSystem. Initializes level as 1, xp as 0 and xpNeededForLevelUp as 20.
- void GainXP (float amount)

  Function to Add xp to this LevelUpInstace.
- void LevelUp ()

  Functio to Directly level up this LevelUpInstance.
- int GetLevel ()

  Function to get the level of this LevelUpInstance.
- float GetHPModifier ()

  Function to get HP modifier of this LevelUpInstance.

7.18.1 Constructor & Destructor Documentation

7.18.1.1 LevelUpInstance()

```
LevelUpInstance::LevelUpInstance ( )
```

Construct a new Level Up Instance object, designed to only be used from the LevelUpSystem. Initializes level as 1, xp as 0 and xpNeededForLevelUp as 20.

7.18.2 Member Function Documentation

7.18.2.1 GainXP()

```
void LevelUpInstance::GainXP (
            float amount )
```

Function to Add xp to this LevelUpInstace.

Parameters

amount	Ammount to of XP add

7.18.2.2 GetHPModifier()

```
float LevelUpInstance::GetHPModifier ( )
```

Function to get HP modifier of this LevelUpInstance.

Returns

float HP modifier of this LevelUpInstance

7.18.2.3 GetLevel()

```
int LevelUpInstance::GetLevel ( )  [inline]
```

Function to get the level of this LevelUpInstance.

Returns

int The level of this LevelUpInstance

7.18.2.4 LevelUp()

```
void LevelUpInstance::LevelUp ( )
```

Functio to Directly level up this LevelUpInstance.

The documentation for this class was generated from the following files:

- src/Utility/LevelUpInstance.hpp
- src/Utility/LevelUpInstance.cpp

7.19 LevelUpSystem Class Reference

```
#include <LevelUpSystem.hpp>
```

Static Public Member Functions

- static void AddCharacter (Character *character)

  Function for adding a character to the LevelUpSystem.
- static void GainXP (Character *character, float amount)

  Function to add XP to a character that is in the LevelUpSystem.
- static void LevelUp (Character *character)

  Function to directly level up a character. Sets character XP to 0.
- static int GetLevel (Character *character)

  Function to get the level of a character in the LevelUpSystem.
- static float GetHPModifier (Character *character)

  Function to get a HP modifier that is calculated from the level of the character. Used to buff character when leveling up.

Static Public Attributes

- static std::unordered_map< Character *, LevelUpInstance > characterLevelMap

7.19.1 Member Function Documentation

7.19.1.1 AddCharacter()

```
void LevelUpSystem::AddCharacter (
            Character * character )  [static]
```

Function for adding a character to the LevelUpSystem.

Parameters

character	Character to add to the LevelUpSystem

7.19.1.2 GainXP()

```
void LevelUpSystem::GainXP (
            Character * character,
            float amount )  [static]
```

Function to add XP to a character that is in the LevelUpSystem.

Parameters

character	Character to add XP to
amount	Amount of XP to add to the character

7.19.1.3 GetHPModifier()

```
float LevelUpSystem::GetHPModifier (
            Character * character )  [static]
```

Function to get a HP modifier that is calculated from the level of the character. Used to buff character when leveling up.

Parameters

character	Character to get HP modifier of

Returns

float The HP modifierr of the character

7.19.1.4 GetLevel()

```
int LevelUpSystem::GetLevel (
            Character * character )  [static]
```

Function to get the level of a character in the LevelUpSystem.

Parameters

character	Character to get level of

Returns

int The level of the character

7.19.1.5 LevelUp()

```
void LevelUpSystem::LevelUp (
            Character * character )  [static]
```

Function to directly level up a character. Sets character XP to 0.

Parameters

character	Character to level up

7.19.2 Member Data Documentation

7.19.2.1 characterLevelMap

```
std::unordered_map< Character *, LevelUpInstance > LevelUpSystem::characterLevelMap  [static]
```

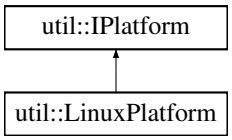
The documentation for this class was generated from the following files:

- src/Utility/LevelUpSystem.hpp
- src/Utility/LevelUpSystem.cpp

7.20 util::LinuxPlatform Struct Reference

`#include <LinuxPlatform.hpp>`

Inheritance diagram for util::LinuxPlatform:



Public Member Functions

- LinuxPlatform ()
- void setIcon (const sf::WindowHandle &inHandle) final
- void toggleFullscreen (const sf::WindowHandle &inHandle, const sf::Uint32 inStyle, const bool inWindowed, const sf::Vector2u &inResolution) final
- float getScreenScalingFactor (const sf::WindowHandle &inHandle) final
- int getRefreshRate (const sf::WindowHandle &inHandle) final

7.20.1 Constructor & Destructor Documentation

7.20.1.1 LinuxPlatform()

```
util::LinuxPlatform::LinuxPlatform ( )
```

7.20.2 Member Function Documentation

7.20.2.1 getRefreshRate()

```
int util::LinuxPlatform::getRefreshRate (
            const sf::WindowHandle & inHandle )  [final], [virtual]
```

Implements util::IPlatform.

7.20.2.2 getScreenScalingFactor()

```
float util::LinuxPlatform::getScreenScalingFactor (
            const sf::WindowHandle & inHandle )  [final], [virtual]
```

Implements util::IPlatform.

7.20.2.3 setIcon()

```
void util::LinuxPlatform::setIcon (
            const sf::WindowHandle & inHandle )  [final], [virtual]
```

Implements util::IPlatform.

7.20.2.4 toggleFullscreen()

```
void util::LinuxPlatform::toggleFullscreen (
            const sf::WindowHandle & inHandle,
            const sf::Uint32 inStyle,
            const bool inWindowed,
            const sf::Vector2u & inResolution )  [final], [virtual]
```

Implements util::IPlatform.

The documentation for this struct was generated from the following file:

- src/Platform/Unix/LinuxPlatform.hpp

7.21 Map Class Reference

```
#include <map.hpp>
```

Public Member Functions

- Map (sf::Vector2u sizeOfRooms, int noRooms, PlayerPS player)
  Initializes variables for new room and calls CreateDungeon()
- ~Map ()
- void RenderCurrentRoom (sf::RenderTarget *target)
  Renders the current room to the given target.
- void CreateDungeon (int noRooms)
  Create the actual dungeon.
- void MovePlayer (Direction dir)
  Moves the player to the room on the map in the given direction.
- RoomInstance * GetRoomInDir (Direction direction)
  Get the Room in the argument direction.
- sf::Vector2i DirToVec (Direction direction)
  Converts a Direction to a unit vector and returns it.
- RoomInstance * GetCurrentRoom ()
  Get the room the player is in on the map.
- RoomInstance * GetSpawnRoom ()
  Get the SpawnRoom.
- void ResetMap ()
- bool IsBossRoomCleared ()
  Checks Whether the bossroom has been cleared.

7.21.1 Constructor & Destructor Documentation

7.21.1.1 Map()

```
Map::Map (
            sf::Vector2u sizeOfRooms,
            int noRooms,
            PlayerPS player )
```

Initializes variables for new room and calls CreateDungeon()

Parameters

sizeOfRooms	the size we want our rooms to be
noRooms	The number of rooms we want the dungeon to consist of

7.21.1.2 ~Map()

```
Map::~Map ( )
```

7.21.2 Member Function Documentation

7.21.2.1 CreateDungeon()

```
void Map::CreateDungeon (
            int noRooms )
```

Create the actual dungeon.

Parameters

noRooms	number of rooms on in the dungeon

7.21.2.2 DirToVec()

```
sf::Vector2i Map::DirToVec (
            Direction direction )
```

Converts a Direction to a unit vector and returns it.

Parameters

direction	the direction

Returns

sf::Vector2i the converted vector

7.21.2.3 GetCurrentRoom()

```
RoomInstance * Map::GetCurrentRoom ( )
```

Get the room the player is in on the map.

Returns

RoomInstance* The room

7.21.2.4 GetRoomInDir()

```
RoomInstance * Map::GetRoomInDir (
            Direction direction )
```

Get the Room in the argument direction.

Parameters

direction	The direction

Returns

returns room, or nullptr if not found

7.21.2.5 GetSpawnRoom()

```
RoomInstance * Map::GetSpawnRoom ( )
```

Get the SpawnRoom.

Returns

RoomInstance* The SpawnRoom

7.21.2.6 IsBossRoomCleared()

```
bool Map::IsBossRoomCleared ( )
```

Checks Whether the bossroom has been cleared.

Returns

true

false

7.21.2.7 MovePlayer()

```
void Map::MovePlayer (
            Direction dir )
```

Moves the player to the room on the map in the given direction.

Parameters

dir	Direction to go

7.21.2.8 RenderCurrentRoom()

```
void Map::RenderCurrentRoom (
            sf::RenderTarget * target )
```

Renders the current room to the given target.

7.21.2.9 ResetMap()

```
void Map::ResetMap ( )
```

The documentation for this class was generated from the following files:

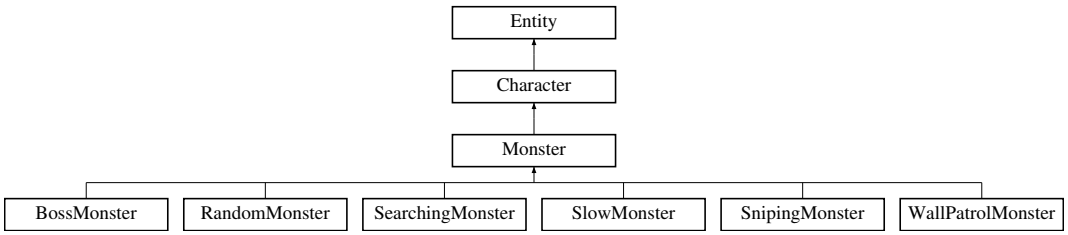
- src/Dungeon/map.hpp
- src/Dungeon/map.cpp

7.22 Monster Class Reference

Monster class, which all our other monsters inherit.

`#include <monster.hpp>`

Inheritance diagram for Monster:



Public Member Functions

- virtual ~Monster ()
- Player & GetPlayer () const
- virtual std::list< ProjectileUP > Attack ()=0
- virtual void Update (float)
- virtual bool Move (float dt)=0
- virtual void Render (sf::RenderTarget *target)
- void initVariables ()
- void SetTarget (PlayerPS target)
- Potion * ReturnPotion ()

  determines if monster is going to drop a potion and what kind of potion.

Protected Member Functions

- Monster (PlayerPS player, sf::Vector2f pos, const std::string &spriteFile)
- Monster (PlayerPS player, float xPos, float yPos, const std::string &spriteFile)
- float getDistanceToPlayer ()
- bool inRangeOfPlayer ()

  Checks if player is in attack range of monster character based on weapon range.
- bool moveTowardsPlayer (float dt)
- void clampPosToRoom ()

  clamps the monsters into a room so they can go into corridors

Protected Attributes

- std::shared_ptr< Player > player_
- sf::RectangleShape healthbar_
- float staticDamage_ = 5.0f
- bool movedLastTick_

Additional Inherited Members

7.22.1 Detailed Description

Monster class, which all our other monsters inherit.

7.22.2 Constructor & Destructor Documentation

7.22.2.1 ~Monster()

```
Monster::~Monster ( )  [virtual]
```

7.22.2.2 Monster() [1/2]

```
Monster::Monster (
            PlayerPS player,
            sf::Vector2f pos,
            const std::string & spriteFile )  [protected]
```

7.22.2.3 Monster() [2/2]

```
Monster::Monster (
            PlayerPS player,
            float xPos,
            float yPos,
            const std::string & spriteFile )  [protected]
```

7.22.3 Member Function Documentation

7.22.3.1 Attack()

```
virtual std::list< ProjectileUP > Monster::Attack ( )  [pure virtual]
```

Implemented in BossMonster, RandomMonster, SearchingMonster, SlowMonster, SnipingMonster, and WallPatrolMonster.

7.22.3.2 clampPosToRoom()

```
void Monster::clampPosToRoom ( )  [protected]
```

clamps the monsters into a room so they can go into corridors

7.22.3.3 getDistanceToPlayer()

```
float Monster::getDistanceToPlayer ( )  [protected]
```

7.22.3.4 GetPlayer()

```
Player & Monster::GetPlayer ( ) const
```

7.22.3.5 initVariables()

```
void Monster::initVariables ( )
```

7.22.3.6 inRangeOfPlayer()

```
bool Monster::inRangeOfPlayer ( )  [protected]
```

Checks if player is in attack range of monster character based on weapon range.

Returns

true if it is

false if not

7.22.3.7 Move()

```
virtual bool Monster::Move (
            float dt )  [pure virtual]
```

Reimplemented from Character.

Implemented in BossMonster, RandomMonster, SearchingMonster, SlowMonster, SnipingMonster, and WallPatrolMonster.

7.22.3.8 moveTowardsPlayer()

```
bool Monster::moveTowardsPlayer (
            float dt )  [protected]
```

7.22.3.9 Render()

```
void Monster::Render (
            sf::RenderTarget * target )  [virtual]
```

Reimplemented from Entity.

7.22.3.10 ReturnPotion()

```
Potion * Monster::ReturnPotion ( )
```

determines if monster is going to drop a potion and what kind of potion.

Returns

Potion*

7.22.3.11 SetTarget()

```
void Monster::SetTarget (
            PlayerPS target )
```

7.22.3.12 Update()

```
void Monster::Update (
            float dt )  [virtual]
```

Implements Character.

7.22.4 Member Data Documentation

7.22.4.1 healthbar_

```
sf::RectangleShape Monster::healthbar_  [protected]
```

7.22.4.2 movedLastTick_

```
bool Monster::movedLastTick_  [protected]
```

7.22.4.3 player_

```
std::shared_ptr<Player> Monster::player_  [protected]
```

7.22.4.4 staticDamage_

```
float Monster::staticDamage_ = 5.0f  [protected]
```

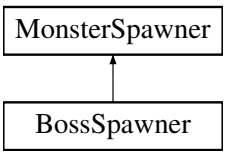
The documentation for this class was generated from the following files:

- src/Actors/Monsters/monster.hpp
- src/Actors/Monsters/monster.cpp

7.23 MonsterSpawner Class Reference

```
#include <MonsterSpawner.hpp>
```

Inheritance diagram for MonsterSpawner:



Public Member Functions

- MonsterSpawner (uint monsterAmount)
- MonsterSpawner ()
- virtual ~MonsterSpawner ()
- virtual MonsterSP SpawnMonster (sf::Vector2u roomSize, PlayerPS target) const
- void SetMonsterAmount (uint amount)
- uint GetMonsterAmount () const

Protected Member Functions

- MonsterSP getRandomMonster (PlayerPS target) const

Protected Attributes

- uint monsterCount_
- uint monsterTypeCount_ = 5

7.23.1 Constructor & Destructor Documentation

7.23.1.1 MonsterSpawner() [1/2]

```
MonsterSpawner::MonsterSpawner (
            uint monsterAmount )  [inline]
```

7.23.1.2 MonsterSpawner() [2/2]

```
MonsterSpawner::MonsterSpawner ( )  [inline]
```

7.23.1.3 ~MonsterSpawner()

```
virtual MonsterSpawner::~MonsterSpawner ( )  [inline], [virtual]
```

7.23.2 Member Function Documentation

7.23.2.1 GetMonsterAmount()

```
uint MonsterSpawner::GetMonsterAmount ( ) const
```

7.23.2.2 getRandomMonster()

```
MonsterSP MonsterSpawner::getRandomMonster (
            PlayerPS target ) const  [protected]
```

7.23.2.3 SetMonsterAmount()

```
void MonsterSpawner::SetMonsterAmount (
            uint amount )
```

7.23.2.4 SpawnMonster()

```
MonsterSP MonsterSpawner::SpawnMonster (
            sf::Vector2u roomSize,
            PlayerPS target ) const  [virtual]
```

7.23.3 Member Data Documentation

7.23.3.1 monsterCount_

```
uint MonsterSpawner::monsterCount_  [protected]
```

7.23.3.2 monsterTypeCount_

```
uint MonsterSpawner::monsterTypeCount_ = 5  [protected]
```

The documentation for this class was generated from the following files:

- src/Actors/Monsters/MonsterSpawner/MonsterSpawner.hpp
- src/Actors/Monsters/MonsterSpawner/MonsterSpawner.cpp

7.24 Collision::OrientedBoundingBox Class Reference

Public Member Functions

- OrientedBoundingBox (const sf::Sprite &Object)
- void ProjectOntoAxis (const sf::Vector2f &Axis, float &Min, float &Max)

Public Attributes

- sf::Vector2f Points [4]

7.24.1 Constructor & Destructor Documentation

7.24.1.1 OrientedBoundingBox()

```
Collision::OrientedBoundingBox::OrientedBoundingBox (
            const sf::Sprite & Object )  [inline]
```

7.24.2 Member Function Documentation

7.24.2.1 ProjectOntoAxis()

```
void Collision::OrientedBoundingBox::ProjectOntoAxis (
            const sf::Vector2f & Axis,
            float & Min,
            float & Max )  [inline]
```

7.24.3 Member Data Documentation

7.24.3.1 Points

```
sf::Vector2f Collision::OrientedBoundingBox::Points[4]
```

The documentation for this class was generated from the following file:

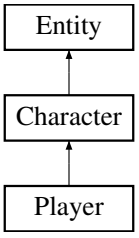
- src/Utility/Collision.cpp

7.25 Player Class Reference

Player class, our controlled character.

```
#include <player.hpp>
```

Inheritance diagram for Player:



Public Member Functions

- Player ()
- ~Player ()
- virtual void Update (float dt)
- void Dash ()

  Handles the dashing functions.
- std::list< ProjectileUP > Attack (sf::Vector2f aimPos)

  handles the attacking of player character
- void initVariables ()
- void ResetDashCooldown ()
- float GetDashCooldownLeft () const
- float GetDashCooldownLength () const
- void AddPotion (Potion *potion)

  Adds a potion to be used later.
- void UsePotion (const std::string &colour)

  Get the colour of the potion player is going to use and runs the funcions to use it.
- std::vector< Potion * > GetInventory () const

  gets inventory of
- void ClearInventory ()

Public Attributes

- bool CanDash
- bool IsDashing

Additional Inherited Members

7.25.1 Detailed Description

Player class, our controlled character.

7.25.2 Constructor & Destructor Documentation

7.25.2.1 Player()

```
Player::Player ( )
```

7.25.2.2 ~Player()

```
Player::~Player ( )  [inline]
```

7.25.3 Member Function Documentation

7.25.3.1 AddPotion()

```
void Player::AddPotion (
            Potion * potion )
```

Adds a potion to be used later.

Parameters

potion	potion to be added to inventory

7.25.3.2 Attack()

```
std::list< ProjectileUP > Player::Attack (
            sf::Vector2f aimPos )
```

handles the attacking of player character

Parameters

aimPos	position of where mouse is aiming

Returns

std::list<ProjectileUP>

7.25.3.3 ClearInventory()

```
void Player::ClearInventory ( )
```

7.25.3.4 Dash()

```
void Player::Dash ( )
```

Handles the dashing functions.

7.25.3.5 GetDashCooldownLeft()

```
float Player::GetDashCooldownLeft ( ) const  [inline]
```

7.25.3.6 GetDashCooldownLength()

```
float Player::GetDashCooldownLength ( ) const  [inline]
```

7.25.3.7 GetInventory()

```
std::vector< Potion * > Player::GetInventory ( ) const
```

gets inventory of

Returns

std::vector<Potion*>

7.25.3.8 initVariables()

```
void Player::initVariables ( )
```

7.25.3.9 ResetDashCooldown()

```
void Player::ResetDashCooldown ( )
```

7.25.3.10 Update()

```
void Player::Update (
            float dt )  [virtual]
```

Implements Character.

7.25.3.11 UsePotion()

```
void Player::UsePotion (
            const std::string & colour )
```

Get the colour of the potion player is going to use and runs the funcions to use it.

Parameters

colour	colour of potion

7.25.4 Member Data Documentation

7.25.4.1 CanDash

```
bool Player::CanDash
```

7.25.4.2 IsDashing

```
bool Player::IsDashing
```

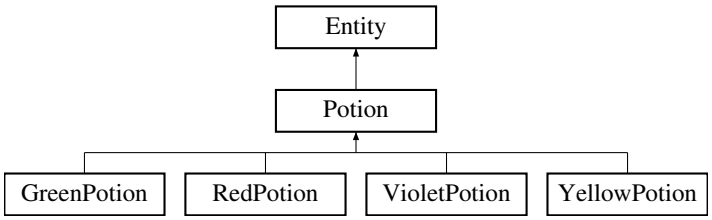
The documentation for this class was generated from the following files:

- src/Actors/player.hpp
- src/Actors/player.cpp

7.26 Potion Class Reference

```
#include <Potion.hpp>
```

Inheritance diagram for Potion:



Public Member Functions

- Potion (const std::string &spritefile, sf::Vector2f pos, int healthIncrease, const std::string &colour)
- virtual ~Potion ()
- const std::string & GetColour () const

  Get the Colour of the potion.
- int GetHealthIncrease () const

  Get the Healing effect of the potion.

Protected Attributes

- int healthIncrease_
- std::string colour_

Additional Inherited Members

7.26.1 Constructor & Destructor Documentation

7.26.1.1 Potion()

```
Potion::Potion (
            const std::string & spritefile,
            sf::Vector2f pos,
            int healthIncrease,
            const std::string & colour )
```

7.26.1.2 ~Potion()

```
virtual Potion::~Potion ( )  [inline], [virtual]
```

7.26.2 Member Function Documentation

7.26.2.1 GetColour()

```
const std::string & Potion::GetColour ( ) const
```

Get the Colour of the potion.

Returns

const refrence to a string with the color

7.26.2.2 GetHealthIncrease()

```
int Potion::GetHealthIncrease ( ) const
```

Get the Healing effect of the potion.

Returns

int

7.26.3 Member Data Documentation

7.26.3.1 colour_

```
std::string Potion::colour_  [protected]
```

7.26.3.2 healthIncrease_

```
int Potion::healthIncrease_  [protected]
```

The documentation for this class was generated from the following files:

- src/Combat/Health/Potion.hpp
- src/Combat/Health/Potion.cpp

7.27 PowerUp Class Reference

```
#include <PowerUp.hpp>
```

Public Member Functions

- PowerUp ()
- ~PowerUp ()

7.27.1 Constructor & Destructor Documentation

7.27.1.1 PowerUp()

```
PowerUp::PowerUp ( )  [inline]
```

7.27.1.2 ~PowerUp()

```
PowerUp::~PowerUp ( )  [inline]
```

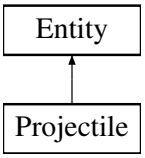
The documentation for this class was generated from the following file:

- src/Combat/PowerUps/PowerUp.hpp

7.28 Projectile Class Reference

`#include <Projectile.hpp>`

Inheritance diagram for Projectile:



Public Types

- enum Type { PlayerProjectile , EnemyProjectile }

Public Member Functions

- Projectile (sf::Sprite &sprite, sf::Vector2f startPos, bool penetratesObjects=false)
- bool HasHit (Character ∗c)

  Function to check if this projectile has already hit a specific character.
- void Hit (Character ∗c)

  Function to add character c to the characters this projectile has hit so it cannot hit it again.
- int GetDamage ()

  Get the Damage that this projectile will make when colliding.
- Projectile::Type GetType ()

  Get the Type of this projectile.
- sf::Vector2f GetDirection ()

  Get the Direction that this Projectile is travelling in.
- sf::Vector2f GetStartPosition ()

  Get the position that this Projectile was spawned at.
- float GetTimeExisted ()

  Get the time in seconds that this Projectile has existed.
- float GetTimeLifeSpan ()

  Get the time in seconds that this Projectile will exist before being deleted.
- float GetProjectileSpeed ()

  Get the speed that this Projectile is travelling at.
- float GetDistanceLifeSpan ()

  Get the distance that this Projectile will travell before being deleted.
- bool Penetrates ()

  Check if this Projectile can penetrate characters or not. Meaning if it can go through characters.
- bool IsAlive ()

  Checks if this Projectile should be deleted or not.
- void SetType (Projectile::Type type)

  Set the Type of this projectile.
- void SetDirection (sf::Vector2f direction)

  Set the Direction of this projectile.
- void SetDamage (int damage)

  Set the Damage of this projectile.

- void SetProjectileSpeed (float projectileSpeed)

  Set the Speed of this projectile.
- void SetTimeLifeSpan (float timeLifeSpan)

  Set the timeLifeSpan of this projectile.
- void SetDistanceLifeSpan (float distanceLifeSpan)

  Set the distanceLifeSpan of this projectile.
- void SetSprite (sf::Sprite sprite)

  Set the Sprite of this Projectile.
- void Kill ()

  Function to make the game delete this Projectile.
- void Update (float dt)

  Function to update this projectile.

Additional Inherited Members

7.28.1 Member Enumeration Documentation

7.28.1.1 Type

```
enum Projectile::Type
```

Enumerator

PlayerProjectile	
EnemyProjectile	

7.28.2 Constructor & Destructor Documentation

7.28.2.1 Projectile()

```
Projectile::Projectile (
            sf::Sprite & sprite,
            sf::Vector2f startPos,
            bool penetratesObjects = false )
```

7.28.3 Member Function Documentation

7.28.3.1 GetDamage()

```
int Projectile::GetDamage ( )  [inline]
```

Get the Damage that this projectile will make when colliding.

Returns

int damage of this projectile

7.28.3.2 GetDirection()

```
sf::Vector2f Projectile::GetDirection ( )  [inline]
```

Get the Direction that this Projectile is travelling in.

Returns

sf::Vector2f direction of this projectile

7.28.3.3 GetDistanceLifeSpan()

```
float Projectile::GetDistanceLifeSpan ( )  [inline]
```

Get the distance that this Projectile will travell before being deleted.

Returns

float distanceLifeSpan of this Projectile

7.28.3.4 GetProjectileSpeed()

```
float Projectile::GetProjectileSpeed ( )  [inline]
```

Get the speed that this Projectile is travelling at.

Returns

float speed of this Projectile

7.28.3.5 GetStartPosition()

```
sf::Vector2f Projectile::GetStartPosition ( )  [inline]
```

Get the position that this Projectile was spawned at.

Returns

sf::Vector2f start postion of this Projectile

7.28.3.6 GetTimeExisted()

```
float Projectile::GetTimeExisted ( )  [inline]
```

Get the time in seconds that this Projectile has existed.

Returns

float timeExisted of this Projectile in seconds

7.28.3.7 GetTimeLifeSpan()

```
float Projectile::GetTimeLifeSpan ( )  [inline]
```

Get the time in seconds that this Projectile will exist before being deleted.

Returns

float timeLifeSpan of this Projectile in seconds

7.28.3.8 GetType()

```
Projectile::Type Projectile::GetType ( )  [inline]
```

Get the Type of this projectile.

Returns

Projectile::Type the type of this projectile

7.28.3.9 HasHit()

```
bool Projectile::HasHit (
            Character * c )
```

Function to check if this projectile has already hit a specific character.

Parameters

c	character to check if this projectile has hit

Returns

true if projectile has hit charachter c

false if projectile has not hit character c

7.28.3.10 Hit()

```
void Projectile::Hit (
            Character * c )
```

Function to add character c to the characters this projectile has hit so it cannot hit it again.

Parameters

c	character to add

7.28.3.11 IsAlive()

```
bool Projectile::IsAlive ( )  [inline]
```

Checks if this Projectile should be deleted or not.

Returns

true Meaning that this Projectile should not be deleted

false Meaning that this Projectile should be deleted

7.28.3.12 Kill()

```
void Projectile::Kill ( )  [inline]
```

Function to make the game delete this Projectile.

7.28.3.13 Penetrates()

```
bool Projectile::Penetrates ( )  [inline]
```

Check if this Projectile can penetrate characters or not. Meaning if it can go through characters.

Returns

true Meaning that this Projectile can penetrate characters

false Meaning that this Projectile can not penetrate characters

7.28.3.14 SetDamage()

```
void Projectile::SetDamage (
            int damage )
```

Set the Damage of this projectile.

Parameters

damage	the new Damage of this projectile

7.28.3.15 SetDirection()

```
void Projectile::SetDirection (
            sf::Vector2f direction )
```

Set the Direction of this projectile.

Parameters

direction	the new Direction of this projectile

7.28.3.16 SetDistanceLifeSpan()

```
void Projectile::SetDistanceLifeSpan (
            float distanceLifeSpan )
```

Set the distanceLifeSpan of this projectile.

Parameters

distanceLifeSpan	the new distanceLifeSpan of this projectile

7.28.3.17 SetProjectileSpeed()

```
void Projectile::SetProjectileSpeed (
            float projectileSpeed )
```

Set the Speed of this projectile.

Parameters

projectileSpeed	the new Speed of this projectile

7.28.3.18 SetSprite()

```
void Projectile::SetSprite (
            sf::Sprite sprite )  [inline]
```

Set the Sprite of this Projectile.

Parameters

sprite	the new Sprite of this projectile

7.28.3.19 SetTimeLifeSpan()

```
void Projectile::SetTimeLifeSpan (
            float timeLifeSpan )
```

Set the timeLifeSpan of this projectile.

Parameters

timeLifeSpan	the new timeLifeSpan of this projectile

7.28.3.20 SetType()

```
void Projectile::SetType (
            Projectile::Type type )
```

Set the Type of this projectile.

Parameters

type	the new type of this projectile

7.28.3.21 Update()

```
void Projectile::Update (
            float dt )
```

Function to update this projectile.

Parameters

dt	deltatime to compensate framerate

The documentation for this class was generated from the following files:

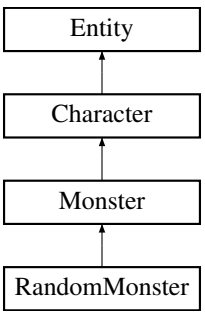
- src/Combat/Projectile.hpp
- src/Combat/projectile.cpp

7.29 RandomMonster Class Reference

RandomMonster is a monster that moves randomly around and shoots projectiles towards the player.

```
#include <RandomMonster.hpp>
```

Inheritance diagram for RandomMonster:



Public Member Functions

- RandomMonster (PlayerPS player, float xPos, float yPos)
- RandomMonster (PlayerPS player, sf::Vector2f pos)
- ~RandomMonster ()
- virtual std::list< ProjectileUP > Attack ()
- virtual bool Move (float dt)
- void initVariables ()

Additional Inherited Members

7.29.1 Detailed Description

RandomMonster is a monster that moves randomly around and shoots projectiles towards the player.

7.29.2 Constructor & Destructor Documentation

7.29.2.1 RandomMonster() [1/2]

```
RandomMonster::RandomMonster (
            PlayerPS player,
            float xPos,
            float yPos )
```

7.29.2.2 RandomMonster() [2/2]

```
RandomMonster::RandomMonster (
            PlayerPS player,
            sf::Vector2f pos )
```

7.29.2.3 ~RandomMonster()

```
RandomMonster::~RandomMonster ( )
```

7.29.3 Member Function Documentation

7.29.3.1 Attack()

```
std::list< ProjectileUP > RandomMonster::Attack ( )  [virtual]
```

Implements Monster.

7.29.3.2 initVariables()

```
void RandomMonster::initVariables ( )
```

7.29.3.3 Move()

```
bool RandomMonster::Move (
            float dt )  [virtual]
```

Implements Monster.

The documentation for this class was generated from the following files:

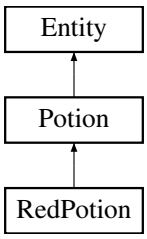
- src/Actors/Monsters/RandomMonster.hpp
- src/Actors/Monsters/RandomMonster.cpp

7.30 RedPotion Class Reference

A red potion. Takes a position as parameter and the colour and healing effect is set automatically.

```
#include <HealthPotions.hpp>
```

Inheritance diagram for RedPotion:



Public Member Functions

- RedPotion (sf::Vector2f pos)

Additional Inherited Members

7.30.1 Detailed Description

A red potion. Takes a position as parameter and the colour and healing effect is set automatically.

7.30.2 Constructor & Destructor Documentation

7.30.2.1 RedPotion()

```
RedPotion::RedPotion (
            sf::Vector2f pos )  [inline]
```

The documentation for this class was generated from the following file:

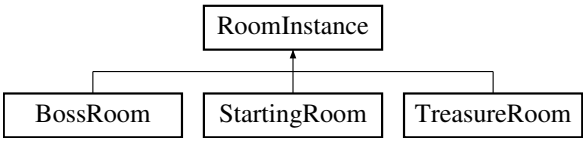
- src/Combat/Health/HealthPotions.hpp

7.31 RoomInstance Class Reference

Class representing a room of a dungeon, usually includes a monsterspawner.

`#include <roomInstance.hpp>`

Inheritance diagram for RoomInstance:



Public Member Functions

- RoomInstance (sf::Vector2u window_size, sf::Vector2i coords)

  Construct a new Room Instance object.
- RoomInstance (sf::Vector2u window_size, sf::Vector2i coords, MonsterSpawner *spawner)
- RoomInstance ()=default
- virtual ~RoomInstance ()
- void Render (sf::RenderTarget *target)

  Renders the room.
- std::vector< RoomTile * > getRoomTilesAt (sf::FloatRect bounds)
- bool positionIsWalkable (sf::FloatRect bounds)
- bool positionIsPenetratable (sf::FloatRect bounds)
- std::vector< std::vector< RoomTile * > > getTiles () const
- void CreateExit (Direction direction)

Takes down a 2-wide in wall (replaces walls with floortiles) to be able to walk between two rooms.

- void Connect (Direction dir, RoomInstance *room)

  Connects two rooms to each other.
- RoomInstance * GetRoomInDir (Direction dir)

  Returns the room found in the wanted direction, nullptr if it is not found.
- sf::Vector2i GetCoords ()

  Get the coordinates of this room on the map.
- void renderSpriteBackground ()

  Renders the rooms background to be a static backdrop, a very expensive operation.
- sf::Vector2u GetEntranceInDirection (Direction direction)
- virtual void Enter (PlayerPS player, Direction direction)

  Function called when entering a room, needs player as parameter to perform some calculations about room difficulty.
- void Exit ()

  Function called when exiting a room modifies cleared_ and visited_ booleans.
- std::vector< MonsterSP > & GetMonsters ()
- void deleteMonster (MonsterSP m)
- Direction RemoveRandomDirection ()

  Function used for Generating the dungeon map. Removes and returns a random Direction from the directionsLeft_ vector.
- void RemoveDirection (Direction dir)

  Function used for Generating the dungeon map. Removes the Direction dir from the directionsLeft_ vector.
- bool HasDirectionsLeft ()

  Function used for Generating the dungeon map. Checks if this roomInstance has any direction that does not already contain an other roomInstance.
- bool IsCleared ()

  Function the check if the roomInstance is cleared of Monsters.
- bool IsVisisted ()
- bool monsterCleared ()
- void AddPotion (Potion *potion)
- std::vector< Potion * > & GetPotions ()

Protected Member Functions

- virtual void setTiles ()

  Helper, Sets all the roomtiles of the roominstance.

Protected Attributes

- sf::Vector2u roomSize_
- sf::Vector2i coords_
- std::vector< std::vector< RoomTile * > > tileVector_
- sf::RenderTexture roomTexture
- sf::Sprite roomBackground
- std::vector< MonsterSP > monsters_
- MonsterSpawner * spawner_
- std::vector< Potion * > potions_
- bool cleared_
- bool visited_
- vector< Direction > directionsLeft_

7.31.1 Detailed Description

Class representing a room of a dungeon, usually includes a monsterspawner.

7.31.2 Constructor & Destructor Documentation

7.31.2.1 RoomInstance() [1/3]

```
RoomInstance::RoomInstance (
            sf::Vector2u window_size,
            sf::Vector2i coords )
```

Construct a new Room Instance object.

Parameters

window_size	TO BE CHANGED takes the windowsize to create a correctly sized room
coords	the map coordinate of the room, follows the cartesian coordinate system

7.31.2.2 RoomInstance() [2/3]

```
RoomInstance::RoomInstance (
            sf::Vector2u window_size,
            sf::Vector2i coords,
            MonsterSpawner * spawner )
```

7.31.2.3 RoomInstance() [3/3]

```
RoomInstance::RoomInstance ( )  [default]
```

7.31.2.4 ~RoomInstance()

```
RoomInstance::~RoomInstance ( )  [virtual]
```

7.31.3 Member Function Documentation

7.31.3.1 AddPotion()

```
void RoomInstance::AddPotion (
            Potion * potion )
```

7.31.3.2 Connect()

```
void RoomInstance::Connect (
            Direction dir,
            RoomInstance * room )
```

Connects two rooms to each other.

Parameters

dir	Direction where the new room resides
room	The new room

Note

Also adds the rooms as a connection on the argument room's side

7.31.3.3 CreateExit()

```
void RoomInstance::CreateExit (
            Direction direction )
```

Takes down a 2-wide in wall (replaces walls with floortiles) to be able to walk between two rooms.

Parameters

direction	The direction of the exit

7.31.3.4 deleteMonster()

```
void RoomInstance::deleteMonster (
            MonsterSP m )
```

7.31.3.5 Enter()

```
void RoomInstance::Enter (
            PlayerPS player,
            Direction direction )  [virtual]
```

Function called when entering a room, needs player as parameter to perform some calculations about room difficulty.

Parameters

player	
direction	the direction we enter from

Reimplemented in BossRoom.

7.31.3.6 Exit()

```
void RoomInstance::Exit ( )
```

Function called when exiting a room modifies cleared_ and visited_ booleans.

7.31.3.7 GetCoords()

```
sf::Vector2i RoomInstance::GetCoords ( )  [inline]
```

Get the coordinates of this room on the map.

Returns

sf::Vector2i

7.31.3.8 GetEntranceInDirection()

```
sf::Vector2u RoomInstance::GetEntranceInDirection (
            Direction direction )
```

7.31.3.9 GetMonsters()

```
std::vector< MonsterSP > & RoomInstance::GetMonsters ( )
```

7.31.3.10 GetPotions()

```
std::vector< Potion * > & RoomInstance::GetPotions ( )
```

7.31.3.11 GetRoomInDir()

```
RoomInstance * RoomInstance::GetRoomInDir (
            Direction dir )
```

Returns the room found in the wanted direction, nullptr if it is not found.

Parameters

dir	The direction

Returns

RoomInstance*

7.31.3.12 getRoomTilesAt()

```
std::vector< RoomTile * > RoomInstance::getRoomTilesAt (
            sf::FloatRect bounds )
```

7.31.3.13 getTiles()

```
std::vector< std::vector< RoomTile * > > RoomInstance::getTiles ( ) const
```

7.31.3.14 HasDirectionsLeft()

```
bool RoomInstance::HasDirectionsLeft ( )
```

Function used for Generating the dungeon map. Checks if this roomInstance has any direction that does not already contain an other roomInstance.

Returns

true Meaning that this roomInstance can be used to generate a new roomInstance

false Meaning that this roomInstance cannot be used to generate a new roomInstance

7.31.3.15 IsCleared()

```
bool RoomInstance::IsCleared ( )
```

Function the check if the roomInstance is cleared of Monsters.

Returns

true Meaning that this roomInstance is cleared of Monsters

false Meaning that this roomInstance is not cleared of Monsters

7.31.3.16 IsVisisted()

```
bool RoomInstance::IsVisisted ( )  [inline]
```

7.31.3.17 monsterCleared()

```
bool RoomInstance::monsterCleared ( )
```

7.31.3.18 positionIsPenetratable()

```
bool RoomInstance::positionIsPenetratable (
            sf::FloatRect bounds )
```

7.31.3.19 positionIsWalkable()

```
bool RoomInstance::positionIsWalkable (
            sf::FloatRect bounds )
```

7.31.3.20 RemoveDirection()

```
void RoomInstance::RemoveDirection (
            Direction dir )
```

Function used for Generating the dungeon map. Removes the Direction dir from the directionsLeft_ vector.

Parameters

dir	The Direction to remove.

7.31.3.21 RemoveRandomDirection()

```
Direction RoomInstance::RemoveRandomDirection ( )
```

Function used for Generating the dungeon map. Removes and returns a random Direction from the directionsLeft_ vector.

Returns

Direction The Direction that was randomly chosen.

7.31.3.22 Render()

```
void RoomInstance::Render (
            sf::RenderTarget * target )
```

Renders the room.

Parameters

target	Where to render the room. About always game window

7.31.3.23 renderSpriteBackground()

```
void RoomInstance::renderSpriteBackground ( )
```

Renders the rooms background to be a static backdrop, a very expensive operation.

7.31.3.24 setTiles()

```
void RoomInstance::setTiles ( )  [protected], [virtual]
```

Helper, Sets all the roomtiles of the roominstance.

7.31.4 Member Data Documentation

7.31.4.1 cleared_

```
bool RoomInstance::cleared_  [protected]
```

7.31.4.2 coords_

```
sf::Vector2i RoomInstance::coords_  [protected]
```

7.31.4.3 directionsLeft_

```
vector<Direction> RoomInstance::directionsLeft_  [protected]
```

7.31.4.4 monsters_

```
std::vector<MonsterSP> RoomInstance::monsters_  [protected]
```

7.31.4.5 potions_

```
std::vector<Potion*> RoomInstance::potions_  [protected]
```

7.31.4.6 roomBackground

```
sf::Sprite RoomInstance::roomBackground  [protected]
```

7.31.4.7 roomSize_

```
sf::Vector2u RoomInstance::roomSize_  [protected]
```

7.31.4.8 roomTexture

```
sf::RenderTexture RoomInstance::roomTexture  [protected]
```

7.31.4.9 spawner_

```
MonsterSpawner* RoomInstance::spawner_  [protected]
```

7.31.4.10 tileVector_

```
std::vector<std::vector<RoomTile*> > RoomInstance::tileVector_  [protected]
```

7.31.4.11 visited_

```
bool RoomInstance::visited_  [protected]
```

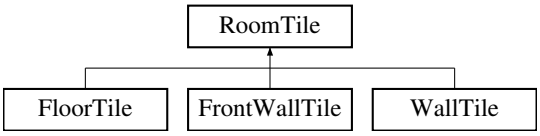
The documentation for this class was generated from the following files:

- src/Dungeon/roomInstance.hpp
- src/Dungeon/roomInstance.cpp

7.32 RoomTile Class Reference

```
#include <roomTile.hpp>
```

Inheritance diagram for RoomTile:



Public Member Functions

- RoomTile (std::string texture, float x, float y, bool walkable, bool penetratable)

  Construct a new Room Tile object.
- const sf::Vector2f getSize () const
- const sf::Vector2f & getPosition () const
- const sf::FloatRect getBoundingBox () const
- const sf::Sprite & getSprite () const
- bool isWalkable () const
- bool isPenetratable () const

7.32.1 Constructor & Destructor Documentation

7.32.1.1 RoomTile()

```
RoomTile::RoomTile (
            std::string texture,
            float x,
            float y,
            bool walkable,
            bool penetratable )
```

Construct a new Room Tile object.

Parameters

texture	texture of the tile
x	location in x-axis
y	location in y-axis
walkable	is it a walkable tile or not
penetratable	can projectiles penetrate the tile

7.32.2 Member Function Documentation

7.32.2.1 getBoundingBox()

```
const sf::FloatRect RoomTile::getBoundingBox ( ) const
```

7.32.2.2 getPosition()

```
const sf::Vector2f & RoomTile::getPosition ( ) const
```

7.32.2.3 getSize()

```
const sf::Vector2f RoomTile::getSize ( ) const
```

7.32.2.4 getSprite()

```
const sf::Sprite & RoomTile::getSprite ( ) const
```

7.32.2.5 isPenetratable()

```
bool RoomTile::isPenetratable ( ) const
```

7.32.2.6 isWalkable()

```
bool RoomTile::isWalkable ( ) const
```

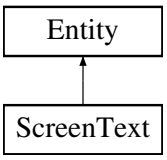
The documentation for this class was generated from the following files:

- src/Dungeon/Tiles/roomTile.hpp
- src/Dungeon/Tiles/roomTile.cpp

7.33 ScreenText Class Reference

```
#include <ScreenText.hpp>
```

Inheritance diagram for ScreenText:



Public Member Functions

- ScreenText (const std::string &textLocation, sf::Vector2f textPos, sf::Vector2f textDims)
- ~ScreenText ()

Additional Inherited Members

7.33.1 Constructor & Destructor Documentation

7.33.1.1 ScreenText()

```
ScreenText::ScreenText (
            const std::string & textLocation,
            sf::Vector2f textPos,
            sf::Vector2f textDims )
```

7.33.1.2 ~ScreenText()

```
ScreenText::~ScreenText ( )  [inline]
```

The documentation for this class was generated from the following files:

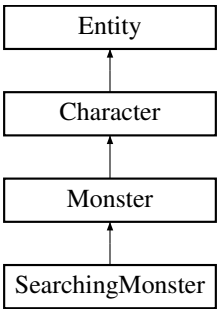
- src/Utility/ScreenText.hpp
- src/Utility/ScreenText.cpp

7.34 SearchingMonster Class Reference

SearchingMonster is our only monster that does not shoot projectile but rather directly decreases the healthpoints of the player. It as its name says always walks towards the player.

```
#include <SearchingMonster.hpp>
```

Inheritance diagram for SearchingMonster:



Public Member Functions

- SearchingMonster (PlayerPS player, float xPos, float yPos)
- SearchingMonster (PlayerPS player, sf::Vector2f pos)
- ~SearchingMonster ()
- virtual std::list< ProjectileUP > Attack ()
- virtual bool Move (float dt)
- void initVariables ()

Additional Inherited Members

7.34.1 Detailed Description

SearchingMonster is our only monster that does not shoot projectile but rather directly decreases the healthpoints of the player. It as its name says always walks towards the player.

7.34.2 Constructor & Destructor Documentation

7.34.2.1 SearchingMonster() [1/2]

```
SearchingMonster::SearchingMonster (
            PlayerPS player,
            float xPos,
            float yPos )
```

7.34.2.2 SearchingMonster() [2/2]

```
SearchingMonster::SearchingMonster (
            PlayerPS player,
            sf::Vector2f pos )
```

7.34.2.3 ~SearchingMonster()

```
SearchingMonster::~SearchingMonster ( )
```

7.34.3 Member Function Documentation

7.34.3.1 Attack()

```
std::list< ProjectileUP > SearchingMonster::Attack ( )  [virtual]
```

Implements Monster.

7.34.3.2 initVariables()

```
void SearchingMonster::initVariables ( )
```

7.34.3.3 Move()

```
bool SearchingMonster::Move (
            float dt )  [virtual]
```

Implements Monster.

The documentation for this class was generated from the following files:

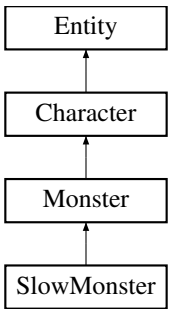
- src/Actors/Monsters/SearchingMonster.hpp
- src/Actors/Monsters/SearchingMonster.cpp

7.35 SlowMonster Class Reference

SlowMonster is a monster that slowly walks towards the player and rapidly shoots a lot of projectiles. The aim of slowmonster is purposfully inacurate.

```
#include <SlowMonster.hpp>
```

Inheritance diagram for SlowMonster:



Public Member Functions

- SlowMonster (PlayerPS player, float xPos, float yPos)
- SlowMonster (PlayerPS player, sf::Vector2f pos)
- ~SlowMonster ()
- virtual std::list< ProjectileUP > Attack ()
- virtual bool Move (float dt)
- void initVariables ()

Additional Inherited Members

7.35.1 Detailed Description

SlowMonster is a monster that slowly walks towards the player and rapidly shoots a lot of projectiles. The aim of slowmonster is purposfully inacurate.

7.35.2 Constructor & Destructor Documentation

7.35.2.1 SlowMonster() [1/2]

```
SlowMonster::SlowMonster (
            PlayerPS player,
            float xPos,
            float yPos )
```

7.35.2.2 SlowMonster() [2/2]

```
SlowMonster::SlowMonster (
            PlayerPS player,
            sf::Vector2f pos )
```

7.35.2.3 ~SlowMonster()

```
SlowMonster::~SlowMonster ( )
```

7.35.3 Member Function Documentation

7.35.3.1 Attack()

```
std::list< ProjectileUP > SlowMonster::Attack ( )  [virtual]
```

Implements Monster.

7.35.3.2 initVariables()

```
void SlowMonster::initVariables ( )
```

7.35.3.3 Move()

```
bool SlowMonster::Move (
            float dt )  [virtual]
```

Implements Monster.

The documentation for this class was generated from the following files:

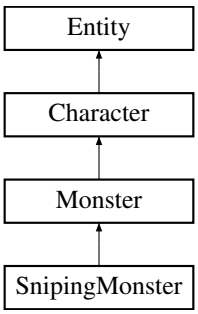
- src/Actors/Monsters/SlowMonster.hpp
- src/Actors/Monsters/SlowMonster.cpp

7.36 SnipingMonster Class Reference

SnipingMonster never moves but shoots projectile towards the player with a really big range.

```
#include <SnipingMonster.hpp>
```

Inheritance diagram for SnipingMonster:



Public Member Functions

- SnipingMonster (PlayerPS player, float xPos, float yPos)
- SnipingMonster (PlayerPS player, sf::Vector2f pos)
- ~SnipingMonster ()
- virtual std::list< ProjectileUP > Attack ()
- virtual bool Move (float dt)
- void initVariables ()

Additional Inherited Members

7.36.1 Detailed Description

SnipingMonster never moves but shoots projectile towards the player with a really big range.

7.36.2 Constructor & Destructor Documentation

7.36.2.1 SnipingMonster() [1/2]

```
SnipingMonster::SnipingMonster (
            PlayerPS player,
            float xPos,
            float yPos )
```

7.36.2.2 SnipingMonster() [2/2]

```
SnipingMonster::SnipingMonster (
            PlayerPS player,
            sf::Vector2f pos )
```

7.36.2.3 ~SnipingMonster()

```
SnipingMonster::~SnipingMonster ( )
```

7.36.3 Member Function Documentation

7.36.3.1 Attack()

```
std::list< ProjectileUP > SnipingMonster::Attack ( )  [virtual]
```

Implements Monster.

7.36.3.2 initVariables()

```
void SnipingMonster::initVariables ( )
```

7.36.3.3 Move()

```
bool SnipingMonster::Move (
            float dt )  [virtual]
```

Implements Monster.

The documentation for this class was generated from the following files:

- src/Actors/Monsters/SnipingMonster.hpp
- src/Actors/Monsters/SnipingMonster.cpp

7.37 SoundEffect Class Reference

```
#include <SoundEffects.hpp>
```

Public Member Functions

- SoundEffect (const std::string &SoundEffectFilename)
- ~SoundEffect ()
- void PlaySound ()

7.37.1 Constructor & Destructor Documentation

7.37.1.1 SoundEffect()

```
SoundEffect::SoundEffect (
            const std::string & SoundEffectFilename )
```

7.37.1.2 ~SoundEffect()

```
SoundEffect::~SoundEffect ( )  [inline]
```

7.37.2 Member Function Documentation

7.37.2.1 PlaySound()

```
void SoundEffect::PlaySound ( )
```

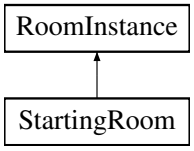
The documentation for this class was generated from the following files:

- src/Utility/Sounds/SoundEffects.hpp
- src/Utility/Sounds/SoundEffects.cpp

7.38 StartingRoom Class Reference

```
#include <startingRoom.hpp>
```

Inheritance diagram for StartingRoom:



Public Member Functions

- StartingRoom (sf::Vector2u window_size, sf::Vector2i choords)

  Construct a new Starting Room object from window size and set its choordinates.
- StartingRoom ()
- ~StartingRoom ()
- virtual void setTiles (sf::Vector2u window_size)

Additional Inherited Members

7.38.1 Constructor & Destructor Documentation

7.38.1.1 StartingRoom() [1/2]

```
StartingRoom::StartingRoom (
            sf::Vector2u window_size,
            sf::Vector2i choords )
```

Construct a new Starting Room object from window size and set its choordinates.

Parameters

window_size	size of window
choords	choordinates of the starting room in map

7.38.1.2 StartingRoom() [2/2]

```
StartingRoom::StartingRoom ( )  [inline]
```

7.38.1.3 ~StartingRoom()

```
StartingRoom::~StartingRoom ( )  [inline]
```

7.38.2 Member Function Documentation

7.38.2.1 setTiles()

```
void StartingRoom::setTiles (
            sf::Vector2u window_size )  [virtual]
```

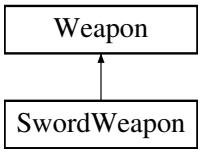
The documentation for this class was generated from the following files:

- src/Dungeon/specialrooms/startingRoom.hpp
- src/Dungeon/specialrooms/startingRoom.cpp

7.39 SwordWeapon Class Reference

```
#include <SwordWeapon.hpp>
```

Inheritance diagram for SwordWeapon:



Public Member Functions

- SwordWeapon (int damage, int range, int rateOfFire, float projectileSpeed, Vector2f projectileSize, const std::string &spriteLocation)
- virtual ~SwordWeapon ()
- virtual ProjectileUP Use (Vector2f dir, Vector2f origin)

Additional Inherited Members

7.39.1 Constructor & Destructor Documentation

7.39.1.1 SwordWeapon()

```
SwordWeapon::SwordWeapon (
            int damage,
            int range,
            int rateOfFire,
            float projectileSpeed,
            Vector2f projectileSize,
            const std::string & spriteLocation )
```

7.39.1.2 ~SwordWeapon()

```
virtual SwordWeapon::~SwordWeapon ( )  [inline], [virtual]
```

7.39.2 Member Function Documentation

7.39.2.1 Use()

```
ProjectileUP SwordWeapon::Use (
            Vector2f dir,
            Vector2f origin )  [virtual]
```

Implements Weapon.

The documentation for this class was generated from the following files:

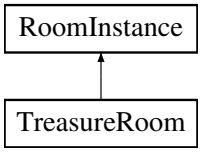
- src/Combat/Weapons/SwordWeapon.hpp
- src/Combat/Weapons/SwordWeapon.cpp

7.40 TreasureRoom Class Reference

Unused, was going to represent a room that includes a treasure and no monsters.

`#include <TreasureRoom.hpp>`

Inheritance diagram for TreasureRoom:



Public Member Functions

- TreasureRoom (sf::Vector2u window_size, sf::Vector2i choords)
- TreasureRoom ()
- ~TreasureRoom ()
- virtual void setTiles (sf::Vector2u window_size)

Additional Inherited Members

7.40.1 Detailed Description

Unused, was going to represent a room that includes a treasure and no monsters.

7.40.2 Constructor & Destructor Documentation

7.40.2.1 TreasureRoom() [1/2]

```
TreasureRoom::TreasureRoom (
            sf::Vector2u window_size,
            sf::Vector2i choords )
```

7.40.2.2 TreasureRoom() [2/2]

```
TreasureRoom::TreasureRoom ( )  [inline]
```

7.40.2.3 ~TreasureRoom()

```
TreasureRoom::~TreasureRoom ( )  [inline]
```

7.40.3 Member Function Documentation

7.40.3.1 setTiles()

```
void TreasureRoom::setTiles (
            sf::Vector2u window_size )  [virtual]
```

The documentation for this class was generated from the following files:

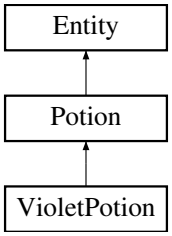
- src/Dungeon/specialrooms/TreasureRoom.hpp
- src/Dungeon/specialrooms/TreasureRoom.cpp

7.41 VioletPotion Class Reference

A violet potion. Takes a position as parameter and the colour and healing effect is set automatically.

`#include <HealthPotions.hpp>`

Inheritance diagram for VioletPotion:



Public Member Functions

- VioletPotion (sf::Vector2f pos)

Additional Inherited Members

7.41.1 Detailed Description

A violet potion. Takes a position as parameter and the colour and healing effect is set automatically.

7.41.2 Constructor & Destructor Documentation

7.41.2.1 VioletPotion()

```
VioletPotion::VioletPotion (
            sf::Vector2f pos )  [inline]
```

The documentation for this class was generated from the following file:

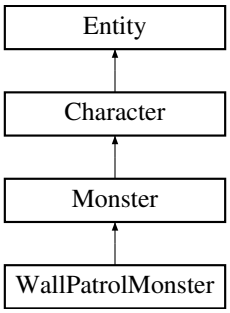
- src/Combat/Health/HealthPotions.hpp

7.42 WallPatrolMonster Class Reference

WallPatrolMonster walks along the walls of the room patroling. Then when the player is in its range it shoots projectiles towards it.

`#include <WallPatrolMonster.hpp>`

Inheritance diagram for WallPatrolMonster:



Public Member Functions

- WallPatrolMonster (PlayerPS player, float xPos, float yPos)
- WallPatrolMonster (PlayerPS player, sf::Vector2f pos)
- ~WallPatrolMonster ()
- virtual std::list< ProjectileUP > Attack ()
- virtual bool Move (float dt)
- void initVariables ()

Additional Inherited Members

7.42.1 Detailed Description

WallPatrolMonster walks along the walls of the room patroling. Then when the player is in its range it shoots projectiles towards it.

7.42.2 Constructor & Destructor Documentation

7.42.2.1 WallPatrolMonster() [1/2]

```
WallPatrolMonster::WallPatrolMonster (
            PlayerPS player,
            float xPos,
            float yPos )
```

7.42.2.2 WallPatrolMonster() [2/2]

```
WallPatrolMonster::WallPatrolMonster (
            PlayerPS player,
            sf::Vector2f pos )
```

7.42.2.3 ~WallPatrolMonster()

```
WallPatrolMonster::~WallPatrolMonster ( )
```

7.42.3 Member Function Documentation

7.42.3.1 Attack()

```
std::list< ProjectileUP > WallPatrolMonster::Attack ( )  [virtual]
```

Implements Monster.

7.42.3.2 initVariables()

```
void WallPatrolMonster::initVariables ( )
```

7.42.3.3 Move()

```
bool WallPatrolMonster::Move (
            float dt )  [virtual]
```

Implements Monster.

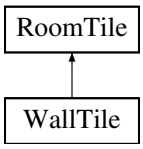
The documentation for this class was generated from the following files:

- src/Actors/Monsters/WallPatrolMonster.hpp
- src/Actors/Monsters/WallPatrolMonster.cpp

7.43 WallTile Class Reference

`#include <roomTile.hpp>`

Inheritance diagram for WallTile:



Public Member Functions

- WallTile (std::string texture, float x, float y)

7.43.1 Constructor & Destructor Documentation

7.43.1.1 WallTile()

```
WallTile::WallTile (
            std::string texture,
            float x,
            float y )  [inline]
```

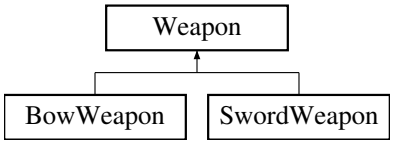
The documentation for this class was generated from the following file:

- src/Dungeon/Tiles/roomTile.hpp

7.44 Weapon Class Reference

`#include <Weapon.hpp>`

Inheritance diagram for Weapon:



Public Member Functions

- Weapon (int damage, int range, int rateOfFire, float projectileSpeed, Vector2f projectileSize, const std::string &spriteLocation)
- virtual ~Weapon ()
- virtual ProjectileUP Use (Vector2f dir, Vector2f origin)=0
- void AddPowerUp (PowerUp *up)
- float GetAttackCooldown ()
- float GetRange ()
- void SetTextureRect (sf::IntRect rect)
- void BoostDamageValue ()
- void UnBoostDamageValue ()

Protected Member Functions

- int getPowerUpCount ()

Protected Attributes

- int defaultDamage_
- int currentDamage_
- float damageBoostModifier = 1.1f
- int range_
- float projectileSpeed_
- Vector2f projectileSize_
- int speed_
- bool penetrates_ = false
- const int maxPowerUps = 3
- Sprite sprite_
- Texture texture_
- vector< PowerUp * > powerUps_
- float cooldown_
- float attackCooldownLength_
- float attackCooldownLeft

7.44.1 Constructor & Destructor Documentation

7.44.1.1 Weapon()

```
Weapon::Weapon (
            int damage,
            int range,
            int rateOfFire,
            float projectileSpeed,
            Vector2f projectileSize,
            const std::string & spriteLocation )
```

7.44.1.2 ~Weapon()

```
virtual Weapon::~Weapon ( )  [inline], [virtual]
```

7.44.2 Member Function Documentation

7.44.2.1 AddPowerUp()

```
void Weapon::AddPowerUp (
            PowerUp * up )
```

7.44.2.2 BoostDamageValue()

```
void Weapon::BoostDamageValue ( )
```

7.44.2.3 GetAttackCooldown()

```
float Weapon::GetAttackCooldown ( )  [inline]
```

7.44.2.4 getPowerUpCount()

```
int Weapon::getPowerUpCount ( )  [protected]
```

7.44.2.5 GetRange()

```
float Weapon::GetRange ( )  [inline]
```

7.44.2.6 SetTextureRect()

```
void Weapon::SetTextureRect (
            sf::IntRect rect )
```

7.44.2.7 UnBoostDamageValue()

```
void Weapon::UnBoostDamageValue ( )
```

7.44.2.8 Use()

```
virtual ProjectileUP Weapon::Use (
            Vector2f dir,
            Vector2f origin )  [pure virtual]
```

Implemented in BowWeapon, and SwordWeapon.

7.44.3 Member Data Documentation

7.44.3.1 attackCooldownLeft

```
float Weapon::attackCooldownLeft  [protected]
```

7.44.3.2 attackCooldownLength_

```
float Weapon::attackCooldownLength_  [protected]
```

7.44.3.3 cooldown_

```
float Weapon::cooldown_  [protected]
```

7.44.3.4 currentDamage_

```
int Weapon::currentDamage_  [protected]
```

7.44.3.5 damageBoostModifier

```
float Weapon::damageBoostModifier = 1.1f  [protected]
```

7.44.3.6 defaultDamage_

```
int Weapon::defaultDamage_  [protected]
```

7.44.3.7 maxPowerUps

```
const int Weapon::maxPowerUps = 3  [protected]
```

7.44.3.8 penetrates_

```
bool Weapon::penetrates_ = false  [protected]
```

7.44.3.9 powerUps_

```
vector<PowerUp*> Weapon::powerUps_  [protected]
```

7.44.3.10 projectileSize_

```
Vector2f Weapon::projectileSize_  [protected]
```

7.44.3.11 projectileSpeed_

```
float Weapon::projectileSpeed_  [protected]
```

7.44.3.12 range_

```
int Weapon::range_  [protected]
```

7.44.3.13 speed_

```
int Weapon::speed_  [protected]
```

7.44.3.14 sprite_

```
Sprite Weapon::sprite_  [protected]
```

7.44.3.15 texture_

```
Texture Weapon::texture_  [protected]
```

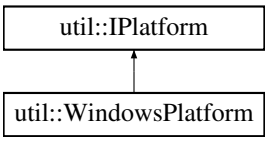
The documentation for this class was generated from the following files:

- src/Combat/Weapons/Weapon.hpp
- src/Combat/Weapons/Weapon.cpp

7.45 util::WindowsPlatform Struct Reference

```
#include <WindowsPlatform.hpp>
```

Inheritance diagram for util::WindowsPlatform:



Public Member Functions

- WindowsPlatform ()
- void setIcon (const sf::WindowHandle &inHandle) final
- void toggleFullscreen (const sf::WindowHandle &inHandle, const sf::Uint32 inStyle, const bool inWindowed, const sf::Vector2u &inResolution) final
- float getScreenScalingFactor (const sf::WindowHandle &inHandle) final
- int getRefreshRate (const sf::WindowHandle &inHandle) final

7.45.1 Constructor & Destructor Documentation

7.45.1.1 WindowsPlatform()

```
util::WindowsPlatform::WindowsPlatform ( )
```

7.45.2 Member Function Documentation

7.45.2.1 getRefreshRate()

```
int util::WindowsPlatform::getRefreshRate (
            const sf::WindowHandle & inHandle )  [final], [virtual]
```

Implements util::IPlatform.

7.45.2.2 getScreenScalingFactor()

```
float util::WindowsPlatform::getScreenScalingFactor (
            const sf::WindowHandle & inHandle )  [final], [virtual]
```

Implements util::IPlatform.

7.45.2.3 setIcon()

```
void util::WindowsPlatform::setIcon (
            const sf::WindowHandle & inHandle )  [final], [virtual]
```

Implements util::IPlatform.

7.45.2.4 toggleFullscreen()

```
void util::WindowsPlatform::toggleFullscreen (
            const sf::WindowHandle & inHandle,
            const sf::Uint32 inStyle,
            const bool inWindowed,
            const sf::Vector2u & inResolution )  [final], [virtual]
```

Implements util::IPlatform.

The documentation for this struct was generated from the following file:

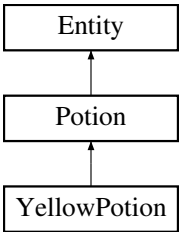
- src/Platform/Win32/WindowsPlatform.hpp

7.46 YellowPotion Class Reference

A yellow potion. Takes a position as parameter and the colour and healing effect is set automatically.

`#include <HealthPotions.hpp>`

Inheritance diagram for YellowPotion:



Public Member Functions

- YellowPotion (sf::Vector2f pos)

Additional Inherited Members

7.46.1 Detailed Description

A yellow potion. Takes a position as parameter and the colour and healing effect is set automatically.

7.46.2 Constructor & Destructor Documentation

7.46.2.1 YellowPotion()

```
YellowPotion::YellowPotion (
            sf::Vector2f pos )  [inline]
```

The documentation for this class was generated from the following file:

- src/Combat/Health/HealthPotions.hpp

Chapter 8

File Documentation

8.1 src/Actors/character.cpp File Reference

```
#include "Actors/character.hpp"
#include "Utility/SpriteHelper.hpp"
```

Macros

- #define C_X 8
- #define C_PIXELS_X 44
- #define C_PIXELS_delta 20
- #define C_PIXELS_Y 64
- #define C_SCALE 2

8.1.1 Macro Definition Documentation

8.1.1.1 C_PIXELS_delta

```
#define C_PIXELS_delta 20
```

8.1.1.2 C_PIXELS_X

```
#define C_PIXELS_X 44
```

8.1.1.3 C_PIXELS_Y

```
#define C_PIXELS_Y 64
```

8.1.1.4 C_SCALE

```
#define C_SCALE 2
```

8.1.1.5 C_X

```
#define C_X 8
```

8.2 src/Actors/character.hpp File Reference

```
#include "Combat/Projectile.hpp"
#include "Animation/Animationhandler.hpp"
#include "Combat/Weapons/Weapon.hpp"
#include "Utility/LevelUpSystem.hpp"
#include "Utility/RandomHelper.hpp"
#include "entity.hpp"
```

Classes

- class Character

  Character class that our player and monster inherits.

Macros

- #define _CHARACTER_CLASS_

8.2.1 Macro Definition Documentation

8.2.1.1 _CHARACTER_CLASS_

```
#define _CHARACTER_CLASS_
```

8.3 character.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #ifndef _CHARACTER_CLASS_
4 #define _CHARACTER_CLASS_
5
6 #include "Combat/Projectile.hpp"
7 //#include "Interfaces/ICollidable.hpp"
8 #include "Animation/Animationhandler.hpp"
9 #include "Combat/Weapons/Weapon.hpp"
10 #include "Utility/LevelUpSystem.hpp"
11 #include "Utility/RandomHelper.hpp"
12 #include "entity.hpp"
17 class Character : public Entity /*, public ICollidable*/ {
18 public:
19     Character(const std::string& filename, sf::Vector2f pos, bool animated = false);
20
21     virtual ~Character();
22
23     virtual void Update(float dt) = 0;
29     void Equip(Weapon* weapon);
30
31     void initVariables();
37     void TakeDamage(int value);
43     void Heal(int value);
44     int GetHitPoints() const;
51     bool IsAlive();
58     bool HasWeapon();
65     bool Idle();
72     bool Dead();
80     bool MoveLeft(float dt);
88     bool MoveRight(float dt);
96     bool MoveDown(float dt);
104     bool MoveUp(float dt);
105     // For subclasses, should be = 0
106     virtual bool Move(float)
107     {
108         return false;
109     }
114     void RevertMove();
115
116     void ResetAttackCooldown();
117     float GetAttackCooldownLeft() const { return attackCooldownLeft; };
118     float GetAttackCooldownLength() const { return attackCooldownLength_; };
119     bool CanAttack;
120     int GetMaxHP();
121
122     void SetNormalSpeed(float value);
123     void ResetCharacterToBeAlive();
124     /*
125     // for ICollidable
126     virtual sf::FloatRect GetBoundingBox() { return sprite_.getGlobalBounds(); }
127     virtual void ProcessCollision(ICollidable* object);
128     virtual ICollidable::EntityType GetEntityType();
129     */
130
131 protected:
132     /*void GetHitBy(Projectile& projectile);*/
133
134     Weapon* weapon_;
135     Projectile::Type characterProjectileType_;
136
137     sf::Vector2f startPos;
138
139     int hitpoints_;
140     int currentMaxHitpoints_;
141     int defaultMaxHitpoints_;
142
143     bool hasAnimation_;
144     AnimationHandler animationHandler_;
145     float currentSpeed_;
146     float defaultSpeed_;
147     bool left_or_right_ = true;
148     void generalUpdate(float dt);
149     bool invincibility_frame_ = false;
150     float attackCooldownLength_;
151     float attackCooldownLeft;
152     void updateAttackCooldown(float dt);
153     std::list<ProjectileUP> emptyList();
154     std::list<ProjectileUP> shotProjectileList(sf::Vector2f aimPos);
155 };
156 #endif
```

8.4 src/Actors/Monsters/BossMonster.cpp File Reference

```
#include "BossMonster.hpp"
#include "math.h"
```

8.5 src/Actors/Monsters/BossMonster.hpp File Reference

```
#include "Actors/character.hpp"
#include "Actors/player.hpp"
#include "Combat/Weapons/SwordWeapon.hpp"
#include "monster.hpp"
```

Classes

- class BossMonster

  This is the boss monster. It is the most powerful monster and once the player kills it the game is won.

Macros

- #define _BOSS_MONSTER_CLASS_

8.5.1 Macro Definition Documentation

8.5.1.1 _BOSS_MONSTER_CLASS_

```
#define _BOSS_MONSTER_CLASS_
```

8.6 BossMonster.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #ifndef _BOSS_MONSTER_CLASS_
4 #define _BOSS_MONSTER_CLASS_
5
6 #include "Actors/character.hpp"
7 #include "Actors/player.hpp"
8 #include "Combat/Weapons/SwordWeapon.hpp"
9 #include "monster.hpp"
10
15 class BossMonster : public Monster {
16 public:
17     BossMonster(PlayerPS player, float xPos, float yPos);
18     BossMonster(PlayerPS player, sf::Vector2f pos);
19     ~BossMonster();
20
21     virtual std::list<ProjectileUP> Attack();
```

```
22      virtual bool Move(float dt);
23
24      void initVariables();
25
26  private:
27      int currentDir_;
28
29      int attackStyle_ = 0;
30      int nofattackStyles_ = 3;
31
32      float angle_ = 0;
33      int nofBulletsInCircle_ = 10;
34      int nofBulletsToShootTowardsPlayer_ = 5;
35      int nofBulletsShot_ = 0;
36
37      sf::Clock attackLoopClock_;
38      float spritalAttackCooldownLength_ = 0.1f;
39      float normalAttackCooldownLength_ = 0.5f;
40
41      float durationUntilTurn_ = 0.5f;
42      float elapsedTurnTime_ = 0.0f;
43
44      void iterateAngle();
45      void iterateAttackStyle();
46  };
47
48  #endif
```

8.7 src/Actors/Monsters/monster.cpp File Reference

```
#include "monster.hpp"
```

8.8 src/Actors/Monsters/monster.hpp File Reference

```
#include "Actors/character.hpp"
#include "Actors/player.hpp"
#include "Combat/Health/HealthPotions.hpp"
```

Classes

- class Monster

  Monster class, which all our other monsters inherit.

Macros

- #define _MONSTER_CLASS_

8.8.1 Macro Definition Documentation

8.8.1.1 _MONSTER_CLASS_

```
#define _MONSTER_CLASS_
```

8.9 monster.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #ifndef _MONSTER_CLASS_
4 #define _MONSTER_CLASS_
5
6 #include "Actors/character.hpp"
7 #include "Actors/player.hpp"
8 #include "Combat/Health/HealthPotions.hpp"
13 class Monster : public Character {
14 public:
15     virtual ~Monster();
16
17     Player& GetPlayer() const;
18     virtual std::list<ProjectileUP> Attack() = 0;
19     virtual void Update(float);
20     virtual bool Move(float dt) = 0;
21     virtual void Render(sf::RenderTarget* target);
22     void initVariables();
23     void SetTarget(PlayerPS target);
29     Potion* ReturnPotion();
30
31 protected:
32     Monster(PlayerPS player, sf::Vector2f pos, const std::string& spriteFile);
33     Monster(PlayerPS player, float xPos, float yPos, const std::string& spriteFile);
34     std::shared_ptr<Player> player_;
35     sf::RectangleShape healthbar_;
36     float staticDamage_ = 5.0f;
37     float getDistanceToPlayer();
44     bool inRangeOfPlayer();
45     bool movedLastTick_;
46
47     bool moveTowardsPlayer(float dt);
52     void clampPosToRoom();
53 };
54
55 #endif
```

8.10 src/Actors/Monsters/MonsterSpawner/BossSpawner.cpp File Reference

```
#include "BossSpawner.hpp"
```

8.11 src/Actors/Monsters/MonsterSpawner/BossSpawner.hpp File Reference

```
#include "../BossMonster.hpp"
#include "MonsterSpawner.hpp"
```

Classes

- class BossSpawner

Macros

- #define _BOSS_SPAWNER_CLASS_

8.11.1 Macro Definition Documentation

8.11.1.1 _BOSS_SPAWNER_CLASS_

```
#define _BOSS_SPAWNER_CLASS_
```

8.12 BossSpawner.hpp

Go to the documentation of this file.

```
1 #pragma once
2 #ifndef _BOSS_SPAWNER_CLASS_
3 #define _BOSS_SPAWNER_CLASS_
4 #include "../BossMonster.hpp"
5 #include "MonsterSpawner.hpp"
6
7 class BossSpawner : public MonsterSpawner {
8 public:
9     BossSpawner()
10         : MonsterSpawner(1)
11     {
12     }
13
14     ~BossSpawner() { }
15     virtual MonsterSP SpawnMonster(sf::Vector2u roomSize, PlayerPS target);
16 };
17
18 #endif
```

8.13 src/Actors/Monsters/MonsterSpawner/MonsterSpawner.cpp File Reference

```
#include "MonsterSpawner.hpp"
#include <time.h>
```

8.14 src/Actors/Monsters/MonsterSpawner/MonsterSpawner.hpp File Reference

```
#include "../BossMonster.hpp"
#include "../RandomMonster.hpp"
#include "../SearchingMonster.hpp"
#include "../SlowMonster.hpp"
#include "../SnipingMonster.hpp"
#include "../WallPatrolMonster.hpp"
```

Classes

- class MonsterSpawner

Macros

- #define _MONSTERSPAWNER_CLASS_

Typedefs

- typedef std::shared_ptr< Monster > MonsterSP
- typedef std::shared_ptr< MonsterSpawner > MonsterSpawnerUP

8.14.1 Macro Definition Documentation

8.14.1.1 _MONSTERSPAWNER_CLASS_

```
#define _MONSTERSPAWNER_CLASS_
```

8.14.2 Typedef Documentation

8.14.2.1 MonsterSP

```
typedef std::shared_ptr<Monster> MonsterSP
```

8.14.2.2 MonsterSpawnerUP

```
typedef std::shared_ptr<MonsterSpawner> MonsterSpawnerUP
```

8.15 MonsterSpawner.hpp

Go to the documentation of this file.

```
1 #pragma once
2 #ifndef _MONSTERSPAWNER_CLASS_
3 #define _MONSTERSPAWNER_CLASS_
4 #include "../BossMonster.hpp"
5 #include "../RandomMonster.hpp"
6 #include "../SearchingMonster.hpp"
7 #include "../SlowMonster.hpp"
8 #include "../SnipingMonster.hpp"
9 #include "../WallPatrolMonster.hpp"
10
11 typedef std::shared_ptr<Monster> MonsterSP;
12
13 class MonsterSpawner {
14
15 public:
16     MonsterSpawner(uint monsterAmount)
17         : monsterCount_(monsterAmount)
18     {
19     }
20     MonsterSpawner()
21         : monsterCount_(0) {};
22
23     virtual ~MonsterSpawner() { }
24
25     // std::list<Monster*> SpawnMonsters(sf::Vector2u roomSize) const;
26     virtual MonsterSP SpawnMonster(sf::Vector2u roomSize, PlayerPS target) const;
27     void SetMonsterAmount(uint amount);
28     uint GetMonsterAmount() const;
29
30 protected:
31     MonsterSP getRandomMonster(PlayerPS target) const;
32     uint monsterCount_;
33     uint monsterTypeCount_ = 5; // update this when adding monsters
34 };
35
36 typedef std::shared_ptr<MonsterSpawner> MonsterSpawnerUP;
37
38 #endif
```

8.16 src/Actors/Monsters/RandomMonster.cpp File Reference

```
#include "RandomMonster.hpp"
```

8.17 src/Actors/Monsters/RandomMonster.hpp File Reference

```
#include "Actors/character.hpp"
#include "Actors/player.hpp"
#include "Combat/Weapons/SwordWeapon.hpp"
#include "monster.hpp"
```

Classes

- class RandomMonster

  RandomMonster is a monster that moves randomly around and shoots projectiles towards the player.

Macros

- #define _RANDOM_MONSTER_CLASS_

8.17.1 Macro Definition Documentation

8.17.1.1 _RANDOM_MONSTER_CLASS_

```
#define _RANDOM_MONSTER_CLASS_
```

8.18 RandomMonster.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #ifndef _RANDOM_MONSTER_CLASS_
4 #define _RANDOM_MONSTER_CLASS_
5
6 #include "Actors/character.hpp"
7 #include "Actors/player.hpp"
8 #include "Combat/Weapons/SwordWeapon.hpp"
9 #include "monster.hpp"
10
15 class RandomMonster : public Monster {
16 public:
17     RandomMonster(PlayerPS player, float xPos, float yPos);
18     RandomMonster(PlayerPS player, sf::Vector2f pos);
19     ~RandomMonster();
20
21     virtual std::list<ProjectileUP> Attack();
22     virtual bool Move(float dt);
23
24     void initVariables();
25
26 private:
27     int currentDir_;
28     float durationUntilTurn_ = 0.5f;
29     float elapsedTurnTime_ = 0.0f;
30 };
31
32 #endif
```

8.19 src/Actors/Monsters/SearchingMonster.cpp File Reference

```
#include "SearchingMonster.hpp"
```

8.20 src/Actors/Monsters/SearchingMonster.hpp File Reference

```
#include "Actors/character.hpp"
#include "Actors/player.hpp"
#include "monster.hpp"
```

Classes

- class SearchingMonster

  SearchingMonster is our only monster that does not shoot projectile but rather directly decreases the healthpoints of the player. It as its name says always walks towards the player.

Macros

- #define _SEARCHING_MONSTER_CLASS_

8.20.1 Macro Definition Documentation

8.20.1.1 _SEARCHING_MONSTER_CLASS_

```
#define _SEARCHING_MONSTER_CLASS_
```

8.21 SearchingMonster.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #ifndef _SEARCHING_MONSTER_CLASS_
4 #define _SEARCHING_MONSTER_CLASS_
5
6 #include "Actors/character.hpp"
7 #include "Actors/player.hpp"
8 #include "monster.hpp"
9
14 class SearchingMonster : public Monster {
15 public:
16     SearchingMonster(PlayerPS player, float xPos, float yPos);
17     SearchingMonster(PlayerPS player, sf::Vector2f pos);
18     ~SearchingMonster();
19
20     virtual std::list<ProjectileUP> Attack();
21     virtual bool Move(float dt);
22
23     void initVariables();
24
25 private:
26     sf::Clock cooldown_;
27     std::string name_ = "Sir Chi";
28 };
29
30 #endif
```

8.22 src/Actors/Monsters/SlowMonster.cpp File Reference

```
#include "SlowMonster.hpp"
```

8.23 src/Actors/Monsters/SlowMonster.hpp File Reference

```
#include "Actors/character.hpp"
#include "Actors/player.hpp"
#include "Combat/Weapons/BowWeapon.hpp"
#include "monster.hpp"
```

Classes

- class SlowMonster

  SlowMonster is a monster that slowly walks towards the player and rapidly shoots a lot of projectiles. The aim of slowmonster is purposfully inacurate.

Macros

- #define _SLOW_MONSTER_CLASS_

8.23.1 Macro Definition Documentation

8.23.1.1 _SLOW_MONSTER_CLASS_

```
#define _SLOW_MONSTER_CLASS_
```

8.24 SlowMonster.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #ifndef _SLOW_MONSTER_CLASS_
4 #define _SLOW_MONSTER_CLASS_
5
6 #include "Actors/character.hpp"
7 #include "Actors/player.hpp"
8 #include "Combat/Weapons/BowWeapon.hpp"
9 #include "monster.hpp"
10
15 class SlowMonster : public Monster {
16 public:
17     SlowMonster(PlayerPS player, float xPos, float yPos);
18     SlowMonster(PlayerPS player, sf::Vector2f pos);
19     ~SlowMonster();
20
21     virtual std::list<ProjectileUP> Attack();
22     virtual bool Move(float dt);
23
24     void initVariables();
25
26 private:
27 };
28
29 #endif
```

8.25 src/Actors/Monsters/SnipingMonster.cpp File Reference

```
#include "SnipingMonster.hpp"
```

8.26 src/Actors/Monsters/SnipingMonster.hpp File Reference

```
#include "Actors/character.hpp"
#include "Actors/player.hpp"
#include "Combat/Weapons/BowWeapon.hpp"
#include "monster.hpp"
```

Classes

- class SnipingMonster

  SnipingMonster never moves but shoots projectile towards the player with a really big range.

Macros

- #define _SNIPING_MONSTER_CLASS_

8.26.1 Macro Definition Documentation

8.26.1.1 _SNIPING_MONSTER_CLASS_

```
#define _SNIPING_MONSTER_CLASS_
```

8.27 SnipingMonster.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #ifndef _SNIPING_MONSTER_CLASS_
4 #define _SNIPING_MONSTER_CLASS_
5
6 #include "Actors/character.hpp"
7 #include "Actors/player.hpp"
8 #include "Combat/Weapons/BowWeapon.hpp"
9 #include "monster.hpp"
10
15 class SnipingMonster : public Monster {
16 public:
17     SnipingMonster(PlayerPS player, float xPos, float yPos);
18     SnipingMonster(PlayerPS player, sf::Vector2f pos);
19     ~SnipingMonster();
20
21     virtual std::list<ProjectileUP> Attack();
22     virtual bool Move(float dt);
23
24     void initVariables();
25
26 private:
27 };
28
29 #endif
```

8.28 src/Actors/Monsters/WallPatrolMonster.cpp File Reference

```
#include "WallPatrolMonster.hpp"
```

8.29 src/Actors/Monsters/WallPatrolMonster.hpp File Reference

```
#include "Actors/character.hpp"
#include "Actors/player.hpp"
#include "Combat/Weapons/SwordWeapon.hpp"
#include "monster.hpp"
```

Classes

- class WallPatrolMonster

  WallPatrolMonster walks along the walls of the room patroling. Then when the player is in its range it shoots projectiles towards it.

Macros

- #define _WALL_PATROL_MONSTER_CLASS_

8.29.1 Macro Definition Documentation

8.29.1.1 _WALL_PATROL_MONSTER_CLASS_

```
#define _WALL_PATROL_MONSTER_CLASS_
```

8.30 WallPatrolMonster.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #ifndef _WALL_PATROL_MONSTER_CLASS_
4 #define _WALL_PATROL_MONSTER_CLASS_
5
6 #include "Actors/character.hpp"
7 #include "Actors/player.hpp"
8 #include "Combat/Weapons/SwordWeapon.hpp"
9 #include "monster.hpp"
10
15 class WallPatrolMonster : public Monster {
16 public:
17     WallPatrolMonster(PlayerPS player, float xPos, float yPos);
18     WallPatrolMonster(PlayerPS player, sf::Vector2f pos);
19     ~WallPatrolMonster();
20
21     virtual std::list<ProjectileUP> Attack();
22     virtual bool Move(float dt);
23
24     void initVariables();
25
26 private:
27     int currentDir_ = 1;
28     void changeDirection();
29 };
30
31 #endif
```

8.31 src/Actors/player.cpp File Reference

```
#include "Actors/player.hpp"
```

8.32 src/Actors/player.hpp File Reference

```
#include "Actors/character.hpp"
#include "Combat/Health/Potion.hpp"
```

Classes

- class Player

  Player class, our controlled character.

Macros

- #define _PLAYER_CLASS_

Typedefs

- typedef std::shared_ptr< Player > PlayerPS

8.32.1 Macro Definition Documentation

8.32.1.1 _PLAYER_CLASS_

```
#define _PLAYER_CLASS_
```

8.32.2 Typedef Documentation

8.32.2.1 PlayerPS

```
typedef std::shared_ptr<Player> PlayerPS
```

8.33 player.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #ifndef _PLAYER_CLASS_
4 #define _PLAYER_CLASS_
5
6 #include "Actors/character.hpp"
7 #include "Combat/Health/Potion.hpp"
12 class Player : public Character {
13 public:
14     Player();
15     ~Player() {};
16
17     virtual void Update(float dt);
22     void Dash();
23
30     std::list<ProjectileUP> Attack(sf::Vector2f aimPos);
31
32     void initVariables();
33
34     void ResetDashCooldown();
35
36     float GetDashCooldownLeft() const { return dashCooldownLeft_; };
37     float GetDashCooldownLength() const { return dashCooldownLength_; };
38
39     bool CanDash;
40     bool IsDashing;
46     void AddPotion(Potion* potion);
52     void UsePotion(const std::string& colour);
58     std::vector<Potion*> GetInventory() const;
59     void ClearInventory();
60
61 private:
62     int attacksBoosted_;
63     int dashesBoosted_;
64
65     float dashSpeed_;
66     float dashLengthBoostModifier = 2.0f;
67     float dashDefaultDurationLength_;
68     float dashCurrentDurationLength_;
69     float dashDurationLeft_;
70     bool deadAnimationPlayed = false;
71     float dt_time = 0;
72     float dashCooldownLength_;
73     float dashCooldownLeft_;
74     void updateDashCooldown(float dt);
75
76     std::vector<Potion*> inventory_;
77 };
78 typedef std::shared_ptr<Player> PlayerPS;
79
80 #endif
```

8.34 src/Animation/animation.cpp File Reference

```
#include "animation.hpp"
#include "Utility/SpriteHelper.hpp"
```

8.35 src/Animation/animation.hpp File Reference

Classes

- class Animation

  Animation build the characters animation frames from spritesheet and displays them to achive an animated sprite.

Macros

- #define _ANIMATION_

8.35.1 Macro Definition Documentation

8.35.1.1 _ANIMATION_

```
#define _ANIMATION_
```

8.36 animation.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #ifndef _ANIMATION_
4 #define _ANIMATION_
6 class Animation {
7 public:
8     Animation() = default;
18     Animation(int x, int y, int width, int height, int spacing, const std::string& textureName);
19     ~Animation() { }
25     void AnimationToSprite(sf::Sprite& sprite) const;
31     void Update(float dt);
32
33 private:
34     static constexpr int nFrames_ = 5;
35     static constexpr float holdTime_ = 0.15f;
36     sf::Texture texture_;
37     sf::IntRect frames_[nFrames_];
38     int iFrame_ = 0;
39     float time_ = 0.0f;
44     void NextFrame();
45 };
46
47 #endif
```

8.37 src/Animation/Animationhandler.cpp File Reference

```
#include "Animationhandler.hpp"
```

8.38 src/Animation/Animationhandler.hpp File Reference

```
#include "animation.hpp"
```

Classes

- class AnimationHandler

Macros

- #define _ANIMATIONHANDLER_CLASS_

Typedefs

- typedef std::shared_ptr< Animation > AnimationPS

Enumerations

- enum AnimationIndex {
  AnimationUp , AnimationDown , AnimationLeft , AnimationRight ,
  AnimationIdle , AnimationDeath , Count }

8.38.1 Macro Definition Documentation

8.38.1.1 _ANIMATIONHANDLER_CLASS_

```
#define _ANIMATIONHANDLER_CLASS_
```

8.38.2 Typedef Documentation

8.38.2.1 AnimationPS

```
typedef std::shared_ptr<Animation> AnimationPS
```

8.38.3 Enumeration Type Documentation

8.38.3.1 AnimationIndex

```
enum AnimationIndex
```

Enumerator

AnimationUp	
AnimationDown	
AnimationLeft	
AnimationRight	
AnimationIdle	
AnimationDeath	
Count	

8.39 Animationhandler.hpp

Go to the documentation of this file.

```
1 #pragma once
2 #ifndef _ANIMATIONHANDLER_CLASS_
3 #define _ANIMATIONHANDLER_CLASS_
4 #include "animation.hpp"
5
6 typedef std::shared_ptr<Animation> AnimationPS;
7
8 enum AnimationIndex {
9     AnimationUp,
10     AnimationDown,
11     AnimationLeft,
12     AnimationRight,
13     AnimationIdle,
14     AnimationDeath,
15     Count
16 };
17
18 class AnimationHandler {
19
20 public:
21     AnimationHandler()
22         : animations_(0) {};
23     ~AnimationHandler();
24     AnimationHandler(uint xOffset, uint yOffset, uint width, uint height, uint xSpacing, const
       std::string& textureLocation, const std::string& deathTexture);
25     void setAnimation(AnimationIndex index);
26     Animation* getAnimation() const;
27
28 private:
29     std::vector<AnimationPS> animations_;
30     AnimationIndex currentAnimationIndex_ = AnimationIndex::AnimationDown;
31 };
32 #endif
```

8.40 src/Combat/CircularProjectile.hpp File Reference

8.41 CircularProjectile.hpp

Go to the documentation of this file.

8.42 src/Combat/Health/HealthPotions.hpp File Reference

```
#include "Actors/player.hpp"
#include "Potion.hpp"
```

Classes

- class GreenPotion

  A green potion. Takes a position as parameter and the colour and healing effect is set automatically.
- class RedPotion

  A red potion. Takes a position as parameter and the colour and healing effect is set automatically.
- class YellowPotion

  A yellow potion. Takes a position as parameter and the colour and healing effect is set automatically.
- class VioletPotion

  A violet potion. Takes a position as parameter and the colour and healing effect is set automatically.

Macros

- #define _HEALTHPOTIONS_CLASS_

8.42.1 Macro Definition Documentation

8.42.1.1 _HEALTHPOTIONS_CLASS_

```
#define _HEALTHPOTIONS_CLASS_
```

8.43 HealthPotions.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #ifndef _HEALTHPOTIONS_CLASS_
4 #define _HEALTHPOTIONS_CLASS_
5 #include "Actors/player.hpp"
6 #include "Potion.hpp"
7
12 class GreenPotion : public Potion {
13 public:
14     GreenPotion(sf::Vector2f pos)
15         : Potion("content/sprites/potions/green_potion.png", pos, 10, "green")
16     {
17     }
18 };
19
24 class RedPotion : public Potion {
25 public:
26     RedPotion(sf::Vector2f pos)
27         : Potion("content/sprites/potions/red_potion.png", pos, 5, "red")
28     {
29     }
30 };
31
36 class YellowPotion : public Potion {
37 public:
38     YellowPotion(sf::Vector2f pos)
39         : Potion("content/sprites/potions/yellow_potion.png", pos, 15, "yellow")
40     {
41     }
42 };
43
48 class VioletPotion : public Potion {
49 public:
50     VioletPotion(sf::Vector2f pos)
51         : Potion("content/sprites/potions/violet_potion.png", pos, 50, "violet")
52     {
53     }
54 };
55
56 #endif
```

8.44 src/Combat/Health/Potion.cpp File Reference

```
#include "Potion.hpp"
```

8.45 src/Combat/Health/Potion.hpp File Reference

```
#include "entity.hpp"
```

Classes

- class Potion

Macros

- #define _POTION_CLASS_

8.45.1 Macro Definition Documentation

8.45.1.1 _POTION_CLASS_

```
#define _POTION_CLASS_
```

8.46 Potion.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #ifndef _POTION_CLASS_
4 #define _POTION_CLASS_
5 #include "entity.hpp"
6
7 class Potion : public Entity {
8 public:
9     Potion(const std::string& spritefile, sf::Vector2f pos, int healthIncrease, const std::string&
       colour);
10
11     virtual ~Potion() {};
12
18     const std::string& GetColour() const;
19
25     int GetHealthIncrease() const;
26
27 protected:
28     int healthIncrease_;
29     std::string colour_;
30 };
31
32 #endif
```

8.47 src/Combat/PowerUps/PowerUp.hpp File Reference

Classes

- class PowerUp

Macros

- #define _POWERUP_CLASS_

8.47.1 Macro Definition Documentation

8.47.1.1 _POWERUP_CLASS_

```
#define _POWERUP_CLASS_
```

8.48 PowerUp.hpp

Go to the documentation of this file.

```
1 #pragma once
2 #ifndef _POWERUP_CLASS_
3 #define _POWERUP_CLASS_
4
5 class PowerUp {
6 private:
7     /* data */
8 public:
9     PowerUp(/* args */) { }
10     ~PowerUp() { }
11 };
12 #endif
```

8.49 src/Combat/projectile.cpp File Reference

```
#include "Combat/Projectile.hpp"
```

8.50 src/Combat/Projectile.hpp File Reference

```
#include "entity.hpp"
```

Classes

- class Projectile

Macros

- #define _Projectile_CLASS_

Typedefs

- typedef std::unique_ptr< Projectile > ProjectileUP

8.50.1 Macro Definition Documentation

8.50.1.1 _Projectile_CLASS_

```
#define _Projectile_CLASS_
```

8.50.2 Typedef Documentation

8.50.2.1 ProjectileUP

```
typedef std::unique_ptr<Projectile> ProjectileUP
```

8.51 Projectile.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #ifndef _Projectile_CLASS_
4 #define _Projectile_CLASS_
5
6 #include "entity.hpp"
7
8 class Character;
9
10 class Projectile : public Entity {
11 public:
12     enum Type { PlayerProjectile,
13         EnemyProjectile };
14
15 public:
16     Projectile(sf::Sprite& sprite, sf::Vector2f startPos, bool penetratesObjects = false);
17
25     bool HasHit(Character* c);
26
32     void Hit(Character* c);
33
39     int GetDamage() { return damage_; }
40
46     Projectile::Type GetType() { return type_; }
47
53     sf::Vector2f GetDirection() { return direction_; }
54
60     sf::Vector2f GetStartPosition() { return startPos_; }
61
67     float GetTimeExisted() { return timeExisted_; }
68
74     float GetTimeLifeSpan() { return timeLifeSpan_; }
75
81     float GetProjectileSpeed() { return projectileSpeed_; }
82
88     float GetDistanceLifeSpan() { return distanceLifeSpanSquared_; }
89
96     bool Penetrates() { return penetrates_; }
```

```
97
104     bool IsAlive() { return alive_; }
105
111     void SetType(Projectile::Type type);
112
118     void SetDirection(sf::Vector2f direction);
119
125     void SetDamage(int damage);
126
132     void SetProjectileSpeed(float projectileSpeed);
133
139     void SetTimeLifeSpan(float timeLifeSpan);
140
146     void SetDistanceLifeSpan(float distanceLifeSpan);
147
153     void SetSprite(sf::Sprite sprite) { sprite_ = sprite; }
154
159     void Kill() { alive_ = false; };
160
166     void Update(float dt);
167
168 private:
169     Type type_;
170     sf::Vector2f direction_;
171     sf::Vector2f startPos_;
172     std::set<Character*> charactersHit_;
173     int damage_;
174     float projectileSpeed_;
175
176     bool alive_;
177     bool penetrates_;
178     float distanceLifeSpanSquared_;
179     float timeExisted_;
180     float timeLifeSpan_;
181
182     void initVariables();
183     bool move(float dt, float x, float y);
184 };
185
186 typedef std::unique_ptr<Projectile> ProjectileUP;
187
188 #endif
```

8.52 src/Combat/Weapons/BowWeapon.cpp File Reference

```
#include "Combat/Weapons/BowWeapon.hpp"
```

8.53 src/Combat/Weapons/BowWeapon.hpp File Reference

```
#include "Combat/Weapons/Weapon.hpp"
```

Classes

- class BowWeapon

Macros

- #define _BOWWEAPON_CLASS_

8.53.1 Macro Definition Documentation

8.53.1.1 _BOWWEAPON_CLASS_

```
#define _BOWWEAPON_CLASS_
```

8.54 BowWeapon.hpp

Go to the documentation of this file.

```
1 #pragma once
2 #ifndef _BOWWEAPON_CLASS_
3 #define _BOWWEAPON_CLASS_
4
5 #include "Combat/Weapons/Weapon.hpp"
6
7 class BowWeapon : public Weapon {
8 private:
9 public:
10      BowWeapon(int damage, int range, int rateOfFire, float projectileSpeed, Vector2f projectileSize,
        const std::string& spriteLocation);
11      virtual ~BowWeapon() { }
12      virtual ProjectileUP Use(Vector2f dir, Vector2f origin);
13 };
14 #endif
```

8.55 src/Combat/Weapons/SwordWeapon.cpp File Reference

```
#include "Combat/Weapons/SwordWeapon.hpp"
```

8.56 src/Combat/Weapons/SwordWeapon.hpp File Reference

```
#include "Combat/Weapons/Weapon.hpp"
```

Classes

- class SwordWeapon

Macros

- #define _SWORDWEAPON_CLASS_

8.56.1 Macro Definition Documentation

8.56.1.1 _SWORDWEAPON_CLASS_

```
#define _SWORDWEAPON_CLASS_
```

8.57 SwordWeapon.hpp

Go to the documentation of this file.

```
1 #pragma once
2 #ifndef _SWORDWEAPON_CLASS_
3 #define _SWORDWEAPON_CLASS_
4
5 #include "Combat/Weapons/Weapon.hpp"
6
7 class SwordWeapon : public Weapon {
8 private:
9 public:
10     SwordWeapon(int damage, int range, int rateOfFire, float projectileSpeed, Vector2f projectileSize,
       const std::string& spriteLocation);
11     virtual ~SwordWeapon() { }
12     virtual ProjectileUP Use(Vector2f dir, Vector2f origin);
13 };
14 #endif
```

8.58 src/Combat/Weapons/Weapon.cpp File Reference

```
#include "Combat/Weapons/Weapon.hpp"
```

8.59 src/Combat/Weapons/Weapon.hpp File Reference

```
#include "Combat/PowerUps/PowerUp.hpp"
#include "Combat/Projectile.hpp"
#include "Utility/SpriteHelper.hpp"
```

Classes

- class Weapon

Macros

- #define _WEAPON_CLASS_

8.59.1 Macro Definition Documentation

8.59.1.1 _WEAPON_CLASS_

```
#define _WEAPON_CLASS_
```

8.60 Weapon.hpp

Go to the documentation of this file.

```
1 #pragma once
2 #ifndef _WEAPON_CLASS_
3 #define _WEAPON_CLASS_
4
5 #include "Combat/PowerUps/PowerUp.hpp"
6 #include "Combat/Projectile.hpp"
7 #include "Utility/SpriteHelper.hpp"
8
9 using namespace std;
10 using namespace sf;
11
12 class Weapon {
13 public:
14     Weapon(int damage, int range, int rateOfFire, float projectileSpeed, Vector2f projectileSize, const
      std::string& spriteLocation);
15     virtual ~Weapon() {};
16     virtual ProjectileUP Use(Vector2f dir, Vector2f origin) = 0;
17     // virtual void Animate() = 0;
18     void AddPowerUp(PowerUp* up);
19     float GetAttackCooldown() { return cooldown_; };
20     float GetRange() { return range_; };
21     void SetTextureRect(sf::IntRect rect);
22     void BoostDamageValue();
23     void UnBoostDamageValue();
24
25 protected:
26     int defaultDamage_;
27     int currentDamage_;
28     float damageBoostModifier = 1.1f;
29
30     int range_;
31     float projectileSpeed_;
32     Vector2f projectileSize_;
33     int speed_;
34     bool penetrates_ = false;
35     const int maxPowerUps = 3;
36     int getPowerUpCount();
37     Sprite sprite_;
38     Texture texture_;
39     vector<PowerUp*> powerUps_;
40     float cooldown_;
41     float attackCooldownLength_;
42     float attackCooldownLeft;
43 };
44 #endif
```

8.61 src/Dungeon/map.cpp File Reference

```
#include "map.hpp"
#include <time.h>
```

8.62 src/Dungeon/map.hpp File Reference

```
#include "Actors/player.hpp"
#include "Dungeon/specialrooms/BossRoom.hpp"
#include "Dungeon/specialrooms/TreasureRoom.hpp"
#include "Dungeon/specialrooms/startingRoom.hpp"
#include "PCH.hpp"
#include "roomInstance.hpp"
```

Classes

- class Map

Macros

- #define _MAP_

8.62.1 Macro Definition Documentation

8.62.1.1 _MAP_

```
#define _MAP_
```

8.63 map.hpp

Go to the documentation of this file.

```
1 #pragma once
2 #ifndef _MAP_
3 #define _MAP_
4 #include "Actors/player.hpp"
5 #include "Dungeon/specialrooms/BossRoom.hpp"
6 #include "Dungeon/specialrooms/TreasureRoom.hpp"
7 #include "Dungeon/specialrooms/startingRoom.hpp"
8 #include "PCH.hpp"
9 #include "roomInstance.hpp"
10
11 class Map {
12
13 public:
20     Map(sf::Vector2u sizeOfRooms, int noRooms, PlayerPS player);
21
22     ~Map();
23
28     void RenderCurrentRoom(sf::RenderTarget* target);
29
35     void CreateDungeon(int noRooms);
36
42     void MovePlayer(Direction dir);
43
50     RoomInstance* GetRoomInDir(Direction direction);
51
58     sf::Vector2i DirToVec(Direction direction);
59
65     RoomInstance* GetCurrentRoom();
66
72     RoomInstance* GetSpawnRoom();
73
74     void ResetMap();
75
82     bool IsBossRoomCleared();
83
84 private:
85     std::pair<int, int> findBossRoom(std::map<std::pair<int, int>, std::set<Direction» coordsAndWalls);
86     void Move(Direction dir);
87
94     RoomInstance* GetRoomAt(sf::Vector2i coord);
95
101     std::pair<int, int> getKey();
102
108     std::pair<int, int> getKey(sf::Vector2i coord);
109     RoomInstance* getRandomRoom();
110     RoomInstance* addRoomToDungeon(sf::Vector2u roomSize, sf::Vector2i coords);
111     void addStartingRoomToDungeon(sf::Vector2u roomSize, sf::Vector2i coords);
112     sf::Vector2u roomSize_;
113     sf::Vector2i currentPos_;
114     PlayerPS player_;
115     sf::Vector2i spawnCoords_;
116     sf::Vector2i bossCoords_;
117     std::map<std::pair<int, int>, RoomInstance*> dungeon_; // cant use vector2i as a key
118     std::vector<std::pair<int, int» existingRoomCoords_;
119 };
120
121 #endif
```

8.64 src/Dungeon/roomInstance.cpp File Reference

```
#include "roomInstance.hpp"
#include <time.h>
```

Namespaces

- namespace direction

Functions

- Direction direction::GetOppositeDir (Direction direction)

8.65 src/Dungeon/roomInstance.hpp File Reference

```
#include "Actors/Monsters/MonsterSpawner/MonsterSpawner.hpp"
#include "Combat/Health/Potion.hpp"
#include "Tiles/roomTile.hpp"
```

Classes

- class RoomInstance

  Class representing a room of a dungeon, usually includes a monsterspawner.

Namespaces

- namespace direction

Enumerations

- enum class Direction {
  Up , Down , Left , Right ,
  Count }

Functions

- Direction direction::GetOppositeDir (Direction direction)

8.65.1 Enumeration Type Documentation

8.65.1.1 Direction

```
enum class Direction  [strong]
```

Enumerator

Up	
Down	
Left	
Right	
Count	

8.66 roomInstance.hpp

Go to the documentation of this file.

```
1
2 #ifndef _ROOM_INSTANCE_
3 #define _ROOM_INSTANCE_
4
5 #include "Actors/Monsters/MonsterSpawner/MonsterSpawner.hpp"
6 #include "Combat/Health/Potion.hpp"
7 #include "Tiles/roomTile.hpp"
8
9 enum class Direction {
10     Up,
11     Down,
12     Left,
13     Right,
14     Count
15 };
16
17 namespace direction {
18 Direction GetOppositeDir(Direction direction);
19 }
20
25 class RoomInstance {
26 public:
33     RoomInstance(sf::Vector2u window_size, sf::Vector2i coords);
34     RoomInstance(sf::Vector2u window_size, sf::Vector2i coords, MonsterSpawner* spawner);
35     RoomInstance() = default;
36     virtual ~RoomInstance();
37
43     void Render(sf::RenderTarget* target);
44
45     std::vector<RoomTile*> getRoomTilesAt(sf::FloatRect bounds);
46
47     bool positionIsWalkable(sf::FloatRect bounds);
48     bool positionIsPenetratable(sf::FloatRect bounds);
49
50     std::vector<std::vector<RoomTile*>> getTiles() const;
51
57     void CreateExit(Direction direction);
58
66     void Connect(Direction dir, RoomInstance* room);
67
74     RoomInstance* GetRoomInDir(Direction dir);
75
81     sf::Vector2i GetCoords() { return coords_; }
82
86     void renderSpriteBackground();
87
88     sf::Vector2u GetEntranceInDirection(Direction direction);
89
96     virtual void Enter(PlayerPS player, Direction direction);
97
102     void Exit();
103
104     std::vector<MonsterSP>& GetMonsters();
105
106     void deleteMonster(MonsterSP m);
107
113     Direction RemoveRandomDirection();
114
120     void RemoveDirection(Direction dir);
121
128     bool HasDirectionsLeft();
129
136     bool IsCleared();
137
```

```
138     bool IsVisisted() {return visited_;}
139
140     bool monsterCleared();
141
142     void AddPotion(Potion* potion);
143
144     std::vector<Potion*>& GetPotions();
145
146 protected:
151     virtual void setTiles();
152     sf::Vector2u roomSize_;
153     sf::Vector2i coords_;
154     std::vector<std::vector<RoomTile*>> tileVector_;
155     sf::RenderTexture roomTexture;
156     sf::Sprite roomBackground;
157     std::vector<MonsterSP> monsters_;
158     MonsterSpawner* spawner_;
159     std::vector<Potion*> potions_;
160     bool cleared_; // whether the room is cleared
161     bool visited_; // whether the room has been visited already
162
163     vector<Direction> directionsLeft_;
164 };
165
166 #endif
```

8.67 src/Dungeon/specialrooms/BossRoom.cpp File Reference

```
#include "BossRoom.hpp"
```

8.68 src/Dungeon/specialrooms/BossRoom.hpp File Reference

```
#include "Actors/Monsters/MonsterSpawner/BossSpawner.hpp"
#include "Dungeon/roomInstance.hpp"
```

Classes

- class BossRoom

  The roominstance where the boss spawns and which has to be cleared to beat the game.

8.69 BossRoom.hpp

Go to the documentation of this file.

```
1 #ifndef _BOSS_ROOM_
2 #define _BOSS_ROOM_
3
4 #include "Actors/Monsters/MonsterSpawner/BossSpawner.hpp"
5 #include "Dungeon/roomInstance.hpp"
6
11 class BossRoom : public RoomInstance {
12 public:
13
18     BossRoom(sf::Vector2u window_size, sf::Vector2i coords);
19     BossRoom() { }
20     ~BossRoom() { }
27     virtual void Enter(PlayerPS player, Direction direction);
28     virtual void setTiles(sf::Vector2u window_size);
29 };
30
31 #endif
```

8.70 src/Dungeon/specialrooms/startingRoom.cpp File Reference

```
#include "startingRoom.hpp"
```

8.71 src/Dungeon/specialrooms/startingRoom.hpp File Reference

```
#include "Dungeon/roomInstance.hpp"
```

Classes

- class StartingRoom

Macros

- #define _STARTING_ROOM_CLASS_

8.71.1 Macro Definition Documentation

8.71.1.1 _STARTING_ROOM_CLASS_

```
#define _STARTING_ROOM_CLASS_
```

8.72 startingRoom.hpp

Go to the documentation of this file.

```
1 #pragma once
2 #ifndef _STARTING_ROOM_CLASS_
3 #define _STARTING_ROOM_CLASS_
4 #include "Dungeon/roomInstance.hpp"
5
6 class StartingRoom : public RoomInstance {
7 public:
14      StartingRoom(sf::Vector2u window_size, sf::Vector2i choords);
15      StartingRoom() { }
16      ~StartingRoom() { }
17      virtual void setTiles(sf::Vector2u window_size);
18
19 private:
20 };
21
22 #endif
```

8.73 src/Dungeon/specialrooms/TreasureRoom.cpp File Reference

```
#include "TreasureRoom.hpp"
```

8.74 src/Dungeon/specialrooms/TreasureRoom.hpp File Reference

```
#include "Dungeon/roomInstance.hpp"
```

Classes

- class TreasureRoom

  Unused, was going to represent a room that includes a treasure and no monsters.

8.75 TreasureRoom.hpp

Go to the documentation of this file.

```
1 #ifndef _TREASURE_ROOM_
2 #define _TREASURE_ROOM_
3
4 #include "Dungeon/roomInstance.hpp"
5
10 class TreasureRoom : public RoomInstance {
11 public:
12     TreasureRoom(sf::Vector2u window_size, sf::Vector2i choords);
13     TreasureRoom() { }
14     ~TreasureRoom() { }
15     virtual void setTiles(sf::Vector2u window_size);
16
17 private:
18 };
19
20 #endif
```

8.76 src/Dungeon/Tiles/roomTile.cpp File Reference

```
#include "roomTile.hpp"
```

8.77 src/Dungeon/Tiles/roomTile.hpp File Reference

```
#include "PCH.hpp"
```

Classes

- class RoomTile
- class WallTile
- class FrontWallTile

  A wall that is not walkable but penetrable.
- class FloorTile

8.78 roomTile.hpp

Go to the documentation of this file.

```
1 #ifndef _ROOM_TILE_
2 #define _ROOM_TILE_
3
4 #include "PCH.hpp"
5
6 class RoomTile {
7 public:
17      RoomTile(std::string texture, float x, float y, bool walkable, bool penetratable);
18
19      const sf::Vector2f getSize() const;
20      const sf::Vector2f& getPosition() const;
21      const sf::FloatRect getBoundingBox() const;
22      const sf::Sprite& getSprite() const;
23      bool isWalkable() const;
24      bool isPenetratable() const;
25
26 private:
27      sf::Vector2f position;
28      sf::Texture tileTexture;
29      sf::Sprite tileSprite;
30      bool walkable_;
31      bool penetratable_;
32
33      bool setTileTexture(std::string);
34 };
35
36 class WallTile : public RoomTile {
37 public:
38      WallTile(std::string texture, float x, float y)
39          : RoomTile(texture, x, y, false, false)
40      {
41      }
42 };
43
48 class FrontWallTile : public RoomTile {
49 public:
50      FrontWallTile(std::string texture, float x, float y)
51          : RoomTile(texture, x, y, false, true)
52      {
53      }
54 };
55
56 class FloorTile : public RoomTile {
57 public:
58      FloorTile(std::string texture, float x, float y)
59          : RoomTile(texture, x, y, true, true)
60      {
61      }
62 };
63
64 #endif
```

8.79 src/entity.cpp File Reference

```
#include "entity.hpp"
```

8.80 src/entity.hpp File Reference

Classes

- class Entity

Macros

- #define _ENTITY_CLASS_

8.80.1 Macro Definition Documentation

8.80.1.1 _ENTITY_CLASS_

```
#define _ENTITY_CLASS_
```

8.81 entity.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #ifndef _ENTITY_CLASS_
4 #define _ENTITY_CLASS_
5 class Entity {
6 public:
15      Entity(const std::string& spriteLocation, float xPos, float yPos, sf::Vector2f spriteDims);
16      Entity(const std::string& spirteLocation, sf::Vector2f pos, sf::Vector2f spriteDims);
17      Entity(sf::Sprite& sprite, float xPos, float yPos);
18      Entity(sf::Sprite& sprite, sf::Vector2f pos);
19
20      virtual ~Entity() {};
21
27      const sf::Sprite& GetSprite() const { return sprite_; };
28
34      const sf::Vector2f& GetPos() const { return pos_; }
35
41      const sf::Vector2i GetPosI() { return sf::Vector2i(pos_); }
42
48      sf::Vector2f GetSpritePosition() const { return sprite_.getPosition(); } // might be unnecessary,
        because sprite pos should be same as that returned of GetPos()
49
55      sf::Vector2f GetSpriteCenter() const;
56
62      const sf::Vector2f& GetOldPosition() const;
63
64      sf::FloatRect GetSpriteBounds() const;
65      sf::FloatRect GetBaseBoxAt(sf::Vector2f pos) const;
66
72      void SetPos(sf::Vector2f pos);
73
79      void SetPosAndOldPos(sf::Vector2f pos);
80
81      virtual void Render(sf::RenderTarget* target);
82
83 protected:
84      void initSprite(const std::string& spriteLocation, sf::Vector2f spriteDims);
85
86      sf::Vector2f pos_;
87      sf::Vector2f oldPos_;
88      sf::Sprite sprite_;
89      sf::Texture texture_;
90 };
91 #endif
```

8.82 src/game.cpp File Reference

```
#include "game.hpp"
#include "Utility/Collision.hpp"
```

Macros

- #define C_PIXELS 64

8.82.1 Macro Definition Documentation

8.82.1.1 C_PIXELS

```
#define C_PIXELS 64
```

8.83 src/game.hpp File Reference

```
#include "Actors/Monsters/BossMonster.hpp"
#include "Actors/player.hpp"
#include "Combat/Health/Potion.hpp"
#include "Combat/Projectile.hpp"
#include "Combat/Weapons/BowWeapon.hpp"
#include "Combat/Weapons/SwordWeapon.hpp"
#include "Dungeon/map.hpp"
#include "Utility/LevelUpSystem.hpp"
#include "Utility/ScreenText.hpp"
#include "Utility/Sounds/SoundEffects.hpp"
#include "gamebar.hpp"
```

Classes

- class Game

Macros

- #define _GAME_CLASS_

8.83.1 Macro Definition Documentation

8.83.1.1 _GAME_CLASS_

```
#define _GAME_CLASS_
```

8.84 game.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #ifndef _GAME_CLASS_
4 #define _GAME_CLASS_
5
6 #include "Actors/Monsters/BossMonster.hpp"
7 #include "Actors/player.hpp"
8 #include "Combat/Health/Potion.hpp"
9 #include "Combat/Projectile.hpp"
10 #include "Combat/Weapons/BowWeapon.hpp"
11 #include "Combat/Weapons/SwordWeapon.hpp"
12 #include "Dungeon/map.hpp"
13 #include "Utility/LevelUpSystem.hpp"
14 #include "Utility/ScreenText.hpp"
15 #include "Utility/Sounds/SoundEffects.hpp"
16 #include "gamebar.hpp"
17
18 class Game {
19 public:
20     Game();
21     ~Game();
26     void UpdateGame();
31     void RenderGame();
38     bool Running() const;
43     void Events();
44
45 private:
46     sf::VideoMode videomode_;
47     sf::RenderWindow* window_;
48     sf::Event event_;
49     sf::Clock dtClock_;
50     PlayerPS player_;
51     Map dungeonMap_;
52     Gamebar gamebar_;
53     ScreenText deathtext_;
54     ScreenText victoryScreen_;
55
56     SoundEffect* playerHitSound = new SoundEffect("content/sounds/playerHit.wav");
57     SoundEffect* monsterHitSound = new SoundEffect("content/sounds/monsterHit.wav");
58
59     float dt_;
60     bool paused = false;
61     bool escapePressedLastTick = paused;
62     bool gameEnder_;
63
64     std::list<ProjectileUP> projectiles_;
65
70     void initWindow();
75     void updateDt();
80     void manageInput();
85     void managePauseInput();
92     void checkCollisions(Character* character, Projectile::Type projectileType);
97     void checkMonsterCollisions();
102     void checkPlayerCollisions();
107     void checkAndHandleProjectileWallCollisions();
113     void deleteProjectile(ProjectileUP p);
119     void addProjectiles(std::list<ProjectileUP> listToAdd);
125     void deleteMonster(Character* m);
126     void deletePotion(Potion* p);
131     void updateProjectiles();
136     void updateMonsters();
144     void updatePotions();
145     bool collidesWithWall(Character* character);
146
154     bool collidesWithWall(Projectile* object);
161     bool ShouldChangeRoom();
168     bool gameLost();
173     void restartGame();
174
181     bool gameWon();
182 };
183
184 #endif
```

8.85 src/gamebar.cpp File Reference

```
#include "gamebar.hpp"
```

8.86 src/gamebar.hpp File Reference

```
#include "Actors/player.hpp"
#include "Combat/Health/HealthPotions.hpp"
#include "Combat/Health/Potion.hpp"
#include "Utility/SpriteHelper.hpp"
```

Classes

- class Gamebar

Macros

- #define _GAMEBAR_CLASS_

8.86.1 Macro Definition Documentation

8.86.1.1 _GAMEBAR_CLASS_

```
#define _GAMEBAR_CLASS_
```

8.87 gamebar.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #ifndef _GAMEBAR_CLASS_
4 #define _GAMEBAR_CLASS_
5
6 #include "Actors/player.hpp"
7 #include "Combat/Health/HealthPotions.hpp"
8 #include "Combat/Health/Potion.hpp"
9 #include "Utility/SpriteHelper.hpp"
10 class Gamebar {
11 public:
17     Gamebar(PlayerPS player);
18
19     Gamebar() {};
25     void Render(sf::RenderTarget* target);
30     void Update();
31
32     void RenderInventory(sf::RenderTarget* target);
33
34 private:
35     PlayerPS player_;
36     sf::Font font_;
37     sf::RectangleShape background_;
38     sf::RectangleShape greenBar_;
39     sf::RectangleShape redBar_;
40     sf::RectangleShape violetBar_;
41     sf::RectangleShape yellowBar_;
42     sf::Text hp_;
43
44     sf::Vector2f redpos_ = sf::Vector2f(800, 5);
45     sf::Vector2f greenpos_ = sf::Vector2f(900, 5);
46     sf::Vector2f yellowpos_ = sf::Vector2f(1000, 5);
```

```
47     sf::Vector2f violetpos_ = sf::Vector2f(1100, 5);
48
49     sf::Sprite redsprite_;
50     sf::Texture redtexture_;
51
52     sf::Sprite greensprite_;
53     sf::Texture greentexture_;
54
55     sf::Sprite yellowsprite_;
56     sf::Texture yellowtexture_;
57
58     sf::Sprite violetsprite_;
59     sf::Texture violettexture_;
60 };
61 #endif
```

8.88 src/Interfaces/CollisionSystem.hpp File Reference

```
#include "Interfaces/ICollidable.hpp"
```

Classes

- class CollisionSystem

  Unused, was not intergrated or completed.

Macros

- #define _COLLISIONSYSTEM_CLASS_

8.88.1 Macro Definition Documentation

8.88.1.1 _COLLISIONSYSTEM_CLASS_

```
#define _COLLISIONSYSTEM_CLASS_
```

8.89 CollisionSystem.hpp

Go to the documentation of this file.

```
1 #pragma once
2 #ifndef _COLLISIONSYSTEM_CLASS_
3 #define _COLLISIONSYSTEM_CLASS_
4 #include "Interfaces/ICollidable.hpp"
5
10 class CollisionSystem {
11 private:
12     // hold all objects that participate in collision
13     std::vector<ICollidable*> objectList;
14
15 public:
16     CollisionSystem() {};
17     void AddToCollisionList(ICollidable* obj);
18     void RemoveFromCollisionList(ICollidable* obj);
19     void ProcessCollisionList();
20 };
21
22 #endif
```

8.90 src/Interfaces/ICollidable.hpp File Reference

Classes

- class ICollidable

  Unused interface for collisionsystem.

Macros

- #define _COLLIDABLE_INTERFACE_

8.90.1 Macro Definition Documentation

8.90.1.1 _COLLIDABLE_INTERFACE_

```
#define _COLLIDABLE_INTERFACE_
```

8.91 ICollidable.hpp

Go to the documentation of this file.

```
1 #pragma once
2 #ifndef _COLLIDABLE_INTERFACE_
3 #define _COLLIDABLE_INTERFACE_
4
9 class ICollidable {
10 public:
11     enum EntityType {
12         character,
13         projectile,
14         tile,
15
16     };
17     // we use these function to retrieve collision relevant information
18     // virtual sf::Vector2f GetPosition() = 0; // objects have position
19     virtual sf::FloatRect GetBoundingBox() = 0; // objects have a size
20
21     // using this function, we notify the object that it collided with something else
22     virtual void ProcessCollision(ICollidable* other) = 0;
23
24     virtual EntityType GetEntityType() = 0;
25
26     virtual ~ICollidable() {};
27 };
28
29 #endif
```

8.92 src/PCH.hpp File Reference

```
#include <SFML/Audio.hpp>
#include <SFML/Graphics.hpp>
#include <SFML/Network.hpp>
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#include <algorithm>
#include <cstdio>
#include <deque>
#include <fstream>
#include <iostream>
#include <list>
#include <map>
#include <memory>
#include <set>
#include <string>
#include <vector>
#include <atomic>
#include <cassert>
#include <cmath>
#include <cstdlib>
#include <exception>
#include <functional>
#include <iomanip>
#include <mutex>
#include <random>
#include <sstream>
#include <thread>
#include <type_traits>
#include "Utility/FileSystem.hpp"
#include "Utility/Types.hpp"
```

Macros

- #define NDEBUG
- #define UNUSED(x) (void)(x)

8.92.1 Macro Definition Documentation

8.92.1.1 NDEBUG

```
#define NDEBUG
```

8.92.1.2 UNUSED

```
#define UNUSED(
              x ) (void)(x)
```

8.93 PCH.hpp

Go to the documentation of this file.

```
1 #ifndef PRECOMPILED_HEADER_HPP
2 #define PRECOMPILED_HEADER_HPP
3
4 #ifndef _DEBUG
5 #ifndef NDEBUG
6 #define NDEBUG
7 #endif
8 #endif // _DEBUG
9
10 // SFML
11 #include <SFML/Audio.hpp>
12 #include <SFML/Graphics.hpp>
13 #include <SFML/Network.hpp>
14 #include <SFML/System.hpp>
15 #include <SFML/Window.hpp>
16
17 // Raspberry Pi
18 #ifdef SFML_SYSTEM_LINUX
19 #ifdef __arm__
20 #define SFML_SYSTEM_PI
21 #endif
22 #endif // SFML SYSTEM_LINUX
23
24 // Typical stdafx.h
25 #include <algorithm>
26 #include <cstdio>
27 #include <deque>
28 #include <fstream>
29 #include <iostream>
30 #include <list>
31 #include <map>
32 #include <memory>
33 #include <set>
34 #include <string>
35 #include <vector>
36
37 // Additional C/C++ libs
38 #include <atomic>
39 #include <cassert>
40 #include <cmath>
41 #include <cstdlib>
42 #include <exception>
43 #include <functional>
44 #include <iomanip>
45 #include <mutex>
46 #include <random>
47 #include <sstream>
48 #include <thread>
49 #include <type_traits>
50
51 // Windows
52 #ifdef _WIN32
53 #ifndef UNICODE
54 #define UNICODE
55 #endif
56
57 #ifndef _UNICODE
58 #define _UNICODE
59 #endif
60
61 #define WIN32_LEAN_AND_MEAN
62 //#include <windows.h>
63 #endif // _WIN32
64
65 // Utils
66 #include "Utility/FileSystem.hpp"
67 #include "Utility/Types.hpp"
68
69 // Macros
70 #define UNUSED(x) (void)(x)
71
72 #endif // PRECOMPILED_HEADER_HPP
```

8.94 src/Platform/IPlatform.hpp File Reference

Classes

- struct util::IPlatform

Namespaces

- namespace util

8.95 IPlatform.hpp

Go to the documentation of this file.

```
1 #ifndef UTIL_IPLATFORM_HELPER_HPP
2 #define UTIL_IPLATFORM_HELPER_HPP
3
4 namespace util
5 {
6     struct IPlatform
7     {
8         virtual ~IPlatform() = default;
9         virtual void setIcon(const sf::WindowHandle &inHandle) = 0;
10         virtual void toggleFullscreen(const sf::WindowHandle &inHandle, const sf::Uint32 inStyle, const
       bool inWindowed, const sf::Vector2u &inResolution) = 0;
11         virtual int getRefreshRate(const sf::WindowHandle &inHandle) = 0;
12         virtual float getScreenScalingFactor(const sf::WindowHandle &inHandle) = 0;
13     };
14 }
15
16 #endif // UTIL_IPLATFORM_HELPER_HPP
```

8.96 src/Platform/Platform.hpp File Reference

Namespaces

- namespace util

8.97 Platform.hpp

Go to the documentation of this file.

```
1 #ifndef UTIL_PLATFORM_HPP
2 #define UTIL_PLATFORM_HPP
3
4 #if defined(__APPLE__)
5     #include "Platform/MacOS/MacOSPlatform.hpp"
6 #elif defined(__linux__)
7     #include "Platform/Unix/LinuxPlatform.hpp"
8 #elif defined(_WIN32)
9     #include "Platform/Win32/WindowsPlatform.hpp"
10 #endif
11
12 namespace util
13 {
14 #if defined(__APPLE__)
15 using Platform = MacOSPlatform;
16 #elif defined(__linux__)
17 using Platform = LinuxPlatform;
18 #elif defined(_WIN32)
19 using Platform = WindowsPlatform;
20 #endif
21 }
22 #endif // UTIL_PLATFORM_HPP
```

8.98 src/Platform/Unix/LinuxPlatform.cpp File Reference

8.99 src/Platform/Unix/LinuxPlatform.hpp File Reference

```
#include "Platform/IPlatform.hpp"
```

Classes

- struct util::LinuxPlatform

Namespaces

- namespace util

8.100 LinuxPlatform.hpp

Go to the documentation of this file.

```
1 #ifndef UTIL_LINUX_PLATFORM_HPP
2 #define UTIL_LINUX_PLATFORM_HPP
3
4 #include "Platform/IPlatform.hpp"
5
6 namespace util
7 {
8 struct LinuxPlatform : IPlatform
9 {
10     LinuxPlatform();
11
12     void setIcon(const sf::WindowHandle& inHandle) final;
13     void toggleFullscreen(const sf::WindowHandle& inHandle, const sf::Uint32 inStyle, const bool
       inWindowed, const sf::Vector2u& inResolution) final;
14     float getScreenScalingFactor(const sf::WindowHandle& inHandle) final;
15     int getRefreshRate(const sf::WindowHandle& inHandle) final;
16 };
17 }
18
19 #endif // UTIL_LINUX_PLATFORM_HPP
```

8.101 src/Platform/Win32/Resource.h File Reference

```
#include <winuser.h>
```

Macros

- #define WIN32_ICON_MAIN 1
- #define MANIFEST_RESOURCE_ID 1

8.101.1 Macro Definition Documentation

8.101.1.1 MANIFEST_RESOURCE_ID

```
#define MANIFEST_RESOURCE_ID 1
```

8.101.1.2 WIN32_ICON_MAIN

```
#define WIN32_ICON_MAIN 1
```

8.102 Resource.h

Go to the documentation of this file.

```
1 #ifndef RESOURCE_H
2 #define RESOURCE_H
3
4 #include <winuser.h>
5
6 #define WIN32_ICON_MAIN 1
7 #define MANIFEST_RESOURCE_ID 1
8
9 #endif // RESOURCE_H
```

8.103 src/Platform/Win32/WindowsPlatform.cpp File Reference

8.104 src/Platform/Win32/WindowsPlatform.hpp File Reference

```
#include "Platform/IPlatform.hpp"
```

Classes

- struct util::WindowsPlatform

Namespaces

- namespace util

8.105 WindowsPlatform.hpp

Go to the documentation of this file.

```
1 #ifndef UTIL_WINDOWS_PLATFORM_HPP
2 #define UTIL_WINDOWS_PLATFORM_HPP
3
4 #include "Platform/IPlatform.hpp"
5
6 // TODO: WM_DISPLAYCHANGE event handling (multi-monitor support)
7
8 namespace util
9 {
10 struct WindowsPlatform : IPlatform
11 {
12     WindowsPlatform();
13
14     void setIcon(const sf::WindowHandle& inHandle) final;
15     void toggleFullscreen(const sf::WindowHandle& inHandle, const sf::Uint32 inStyle, const bool
       inWindowed, const sf::Vector2u& inResolution) final;
16     float getScreenScalingFactor(const sf::WindowHandle& inHandle) final;
17     int getRefreshRate(const sf::WindowHandle& inHandle) final;
18
19 private:
20     PBYTE getIconDirectory(const int inResourceId);
21     HICON getIconFromIconDirectory(PBYTE inIconDirectory, const uint inSize);
22     DWORD sfmlWindowStyleToWin32WindowStyle(const sf::Uint32 inStyle);
23
24     float m_screenScalingFactor = 0.0f;
25     int m_refreshRate = 0;
26 };
27 }
28
29 #endif // UTIL_WINDOWS_PLATFORM_HPP
```

8.106 src/readme.md File Reference

8.107 src/Utility/Collision.cpp File Reference

```
#include "Collision.hpp"
#include "SFML/Graphics.hpp"
#include <map>
```

Classes

- class Collision::BitmaskManager
- class Collision::OrientedBoundingBox

Namespaces

- namespace Collision

Functions

- bool Collision::PixelPerfectTest (const sf::Sprite &Object1, const sf::Sprite &Object2, sf::Uint8 AlphaLimit)
- bool Collision::CreateTextureAndBitmask (sf::Texture &LoadInto, const std::string &Filename)
- sf::Vector2f Collision::GetSpriteCenter (const sf::Sprite &Object)
- sf::Vector2f Collision::GetSpriteSize (const sf::Sprite &Object)
- bool Collision::CircleTest (const sf::Sprite &Object1, const sf::Sprite &Object2)
- bool Collision::BoundingBoxTest (const sf::Sprite &Object1, const sf::Sprite &Object2)

Variables

- BitmaskManager Collision::Bitmasks

8.108 src/Utility/Collision.hpp File Reference

Namespaces

- namespace Collision

Functions

- bool Collision::PixelPerfectTest (const sf::Sprite &Object1, const sf::Sprite &Object2, sf::Uint8 AlphaLimit)
- bool Collision::CreateTextureAndBitmask (sf::Texture &LoadInto, const std::string &Filename)
- bool Collision::CircleTest (const sf::Sprite &Object1, const sf::Sprite &Object2)
- bool Collision::BoundingBoxTest (const sf::Sprite &Object1, const sf::Sprite &Object2)
- sf::Vector2f Collision::GetSpriteCenter (const sf::Sprite &Object)
- sf::Vector2f Collision::GetSpriteSize (const sf::Sprite &Object)

8.109 Collision.hpp

Go to the documentation of this file.

```
1 /*
2  * File:   collision.h
3  * Authors: Nick Koirala (original version), ahnonay (SFML2 compatibility)
4  *
5  * Collision Detection and handling class
6  * For SFML2.
7
8  Notice from the original version:
9
10  (c) 2009 - LittleMonkey Ltd
11
12  This software is provided 'as-is', without any express or
13  implied warranty. In no event will the authors be held
14  liable for any damages arising from the use of this software.
15
16  Permission is granted to anyone to use this software for any purpose,
17  including commercial applications, and to alter it and redistribute
18  it freely, subject to the following restrictions:
19
20  1. The origin of this software must not be misrepresented;
21  you must not claim that you wrote the original software.
22  If you use this software in a product, an acknowledgment
23  in the product documentation would be appreciated but
24  is not required.
25
26  2. Altered source versions must be plainly marked as such,
27  and must not be misrepresented as being the original software.
28
29  3. This notice may not be removed or altered from any
30  source distribution.
31
32  *
33  * Created on 30 January 2009, 11:02
34  */
35
36 #ifndef COLLISION_H
37 #define COLLISION_H
38
39 namespace Collision {
50 bool PixelPerfectTest(const sf::Sprite& Object1, const sf::Sprite& Object2, sf::Uint8 AlphaLimit = 0);
51
59 bool CreateTextureAndBitmask(sf::Texture& LoadInto, const std::string& Filename);
60
66 bool CircleTest(const sf::Sprite& Object1, const sf::Sprite& Object2);
67
72 bool BoundingBoxTest(const sf::Sprite& Object1, const sf::Sprite& Object2);
73
74 sf::Vector2f GetSpriteCenter(const sf::Sprite& Object);
75 sf::Vector2f GetSpriteSize(const sf::Sprite& Object);
76 }
77
78 #endif /* COLLISION_H */
```

8.110 src/Utility/FileSystem.hpp File Reference

```
#include <experimental/filesystem>
```

Namespaces

- namespace util

8.111 FileSystem.hpp

Go to the documentation of this file.

```
1 #ifndef UTIL_FILE_SYSTEM_HPP
2 #define UTIL_FILE_SYSTEM_HPP
3
4 // std::filesystem
5 #if __GNUC__ >= 8 || __clang_major__ >= 9
6     #include <filesystem>
7 #else
8     #include <experimental/filesystem>
9 #endif
10
11 namespace util
12 {
13 #if __GNUC__ >= 8 || __clang_major__ >= 9
14 namespace fs = std::filesystem;
15 #else
16 namespace fs = std::experimental::filesystem::v1;
17 #endif
18 }
19
20 #endif // UTIL_FILE_SYSTEM_HPP
```

8.112 src/Utility/LevelUpInstance.cpp File Reference

```
#include "LevelUpInstance.hpp"
```

8.113 src/Utility/LevelUpInstance.hpp File Reference

Classes

- class LevelUpInstance

Macros

- #define _LevelUpInstance_CLASS_

8.113.1 Macro Definition Documentation

8.113.1.1 _LevelUpInstance_CLASS_

```
#define _LevelUpInstance_CLASS_
```

8.114 LevelUpInstance.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #ifndef _LevelUpInstance_CLASS_
4 #define _LevelUpInstance_CLASS_
5
6 class LevelUpInstance {
7 public:
12      LevelUpInstance();
13
19      void GainXP(float amount);
20
25      void LevelUp();
26
32      int GetLevel() { return level; }
33
39      float GetHPModifier();
40
41 private:
42      int level;
43      float xp;
44      float xpNeededForLevelUp;
45 };
46
47 #endif
```

8.115 src/Utility/LevelUpSystem.cpp File Reference

```
#include "LevelUpSystem.hpp"
```

8.116 src/Utility/LevelUpSystem.hpp File Reference

```
#include "Actors/character.hpp"
#include "LevelUpInstance.hpp"
```

Classes

- class LevelUpSystem

Macros

- #define _LevelUpSystem_CLASS_

8.116.1 Macro Definition Documentation

8.116.1.1 _LevelUpSystem_CLASS_

```
#define _LevelUpSystem_CLASS_
```

8.117 LevelUpSystem.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "Actors/character.hpp"
4 #include "LevelUpInstance.hpp"
5
6 #ifndef _LevelUpSystem_CLASS_
7 #define _LevelUpSystem_CLASS_
8
9 class LevelUpSystem {
10 public:
16     static void AddCharacter(Character* character);
17
24     static void GainXP(Character* character, float amount);
25
31     static void LevelUp(Character* character);
32
39     static int GetLevel(Character* character);
40
47     static float GetHPModifier(Character* character);
48
49     static std::unordered_map<Character*, LevelUpInstance> characterLevelMap;
50 };
51
52 #endif
```

8.118 src/Utility/Main.cpp File Reference

```
#include "Platform/Platform.hpp"
#include "game.hpp"
```

Functions

- int main ()

8.118.1 Function Documentation

8.118.1.1 main()

```
int main ( )
```

8.119 src/Utility/RandomHelper.cpp File Reference

```
#include "Utility/RandomHelper.hpp"
```

Namespaces

- namespace randomhelper

Functions

- float randomhelper::RandomFloatBetween (float min, float max)
- int randomhelper::RandomIntBetween (int min, int max)

8.120 src/Utility/RandomHelper.hpp File Reference

Namespaces

- namespace randomhelper

Macros

- #define _RANDOMHELPER_

Functions

- float randomhelper::RandomFloatBetween (float min, float max)
- int randomhelper::RandomIntBetween (int min, int max)

8.120.1 Macro Definition Documentation

8.120.1.1 _RANDOMHELPER_

```
#define _RANDOMHELPER_
```

8.121 RandomHelper.hpp

Go to the documentation of this file.

```
1 #pragma once
2 #ifndef _RANDOMHELPER_
3 #define _RANDOMHELPER_
4
5 namespace randomhelper {
6 float RandomFloatBetween(float min, float max); //
7 int RandomIntBetween(int min, int max); //Inclusive min and max
8
9 } // namespace
10
11 #endif
```

8.122 src/Utility/ScreenText.cpp File Reference

```
#include "ScreenText.hpp"
```

8.123 src/Utility/ScreenText.hpp File Reference

```
#include "entity.hpp"
```

Classes

- class ScreenText

8.124 ScreenText.hpp

Go to the documentation of this file.

```
1 #ifndef _TEXT_CLASS_
2 #define _TEXT_CLASS_
3
4 #include "entity.hpp"
5
6 class ScreenText : public Entity {
7 public:
8     ScreenText(const std::string& textLocation, sf::Vector2f textPos, sf::Vector2f textDims);
9     ~ScreenText() { }
10
11 private:
12 };
13
14 #endif
```

8.125 src/Utility/Sounds/SoundEffects.cpp File Reference

```
#include "SoundEffects.hpp"
```

8.126 src/Utility/Sounds/SoundEffects.hpp File Reference

Classes

- class SoundEffect

8.127 SoundEffects.hpp

Go to the documentation of this file.

```
1 #ifndef _SOUND_EFFECT_
2 #define _SOUND_EFFECT_
3
4 class SoundEffect {
5 public:
6     SoundEffect(const std::string& SoundEffectFilename);
7     ~SoundEffect() { }
8     void PlaySound();
9
10 private:
11     sf::SoundBuffer buffer;
12     sf::Sound effect;
13 };
14
15 #endif
```

8.128 src/Utility/SpriteHelper.cpp File Reference

```
#include "Utility/SpriteHelper.hpp"
```

Namespaces

- namespace spritehelper

Macros

- #define PI 3.14159265

Functions

- void spritehelper::CreateSpriteFrom (const std::string &spriteFile, sf::Vector2f dimensions, sf::Sprite &sprite, sf::Texture &texture)
- void spritehelper::SetScale (sf::Vector2f wantedDimension, sf::Sprite &sprite)
- void spritehelper::RotateSprite (sf::Vector2f directionOfRotation, sf::Sprite &sprite)
- void spritehelper::SetOriginBottomCenter (sf::Sprite &sprite)

8.128.1 Macro Definition Documentation

8.128.1.1 PI

```
#define PI 3.14159265
```

8.129 src/Utility/SpriteHelper.hpp File Reference

Namespaces

- namespace spritehelper

Macros

- #define _SPRITEHELPER_

Functions

- void spritehelper::CreateSpriteFrom (const std::string &spriteFile, sf::Vector2f dimensions, sf::Sprite &sprite, sf::Texture &texture)
- void spritehelper::SetScale (sf::Vector2f wantedDimension, sf::Sprite &sprite)
- void spritehelper::RotateSprite (sf::Vector2f directionOfRotation, sf::Sprite &sprite)
- void spritehelper::SetOriginBottomCenter (sf::Sprite &sprite)

8.129.1 Macro Definition Documentation

8.129.1.1 _SPRITEHELPER_

```
#define _SPRITEHELPER_
```

8.130 SpriteHelper.hpp

Go to the documentation of this file.

```
1 #pragma once
2 #ifndef _SPRITEHELPER_
3 #define _SPRITEHELPER_
4
5 namespace spritehelper {
6 void CreateSpriteFrom(const std::string& spriteFile, sf::Vector2f dimensions, sf::Sprite& sprite,
      sf::Texture& texture);
7
8 void SetScale(sf::Vector2f wantedDimension, sf::Sprite& sprite);
9
10 void RotateSprite(sf::Vector2f directionOfRotation, sf::Sprite& sprite);
11 void SetOriginBottomCenter(sf::Sprite& sprite);
12
13 } // namespace
14
15 #endif
```

8.131 src/Utility/Types.hpp File Reference

```
#include <cstdint>
```

Typedefs

- typedef std::uint8_t uchar
- typedef std::uint16_t ushort
- typedef std::uint32_t uint
- typedef std::uint64_t ullong
- typedef std::int64_t llong

8.131.1 Typedef Documentation

8.131.1.1 llong

```
typedef std::int64_t llong
```

8.131.1.2 uchar

```
typedef std::uint8_t uchar
```

8.131.1.3 uint

```
typedef std::uint32_t uint
```

8.131.1.4 ullong

```
typedef std::uint64_t ullong
```

8.131.1.5 ushort

```
typedef std::uint16_t ushort
```

8.132 Types.hpp

Go to the documentation of this file.

```
1 #ifndef UTIL_TYPES_HPP
2 #define UTIL_TYPES_HPP
3
4 #include <cstdint>
5
6 typedef std::uint8_t uchar;
7 typedef std::uint16_t ushort;
8 typedef std::uint32_t uint;
9 typedef std::uint64_t ullong;
10
11 typedef std::int64_t llong;
12
13 #endif // UTIL_TYPES_HPP
```

Index